Система автоматического распознавания речи SPIRIT ASR Engine

С. Ю. Иконин, Д. В. Сарана

Статья открывает серию публикаций, посвященных технологии распознавания связной речи и речевых команд и различным аспектам ее практического применения на примере системы SPIRIT ASR Engine, разработанной в компании SPIRIT Corp. Особое внимание при разработке было уделено вопросам устойчивой работы системы в «агрессивной» шумовой обстановке. В данной работе рассматриваются основные принципы построения и применения скрытых марковских моделей (СММ) в системах распознавания речи с ограниченным словарем, а также некоторые практические аспекты реализации системы распознавания слитно произнесенных цепочек слов в режиме реального времени.

Введение

Система распознавания речи SPIRIT ASR Engine разработана для широкого ряда практических задач. К числу таких задач можно отнести, например, организацию автоматического call-центра (голосовое управление системой меню, набор PIN-кода и телефонного номера), системы безопасности, системы речевого управления и т.д. Система способна в режиме реального времени производить дикторонезависимое распознавание цепочек слитно произнесенных слов и отдельных речевых команд, в том числе и в шумовых условиях при соотношении сигнал-шум вплоть до +5дБ.

Качество работы системы было протестировано на речевой базе TIDigits, содержащей цепочки английских цифр от «0» до «9» плюс «Oh». Общее количество высказываний – 8700 (56 мужчин и 57 женщин) при длине строки от 1 до 7 цифр. Точность распознавания изолированных команд (длина строки – 1) составила 99,9%.Точность распознавания цепочек цифр – 97,9%. При использовании информации о длине цепочки качество распознавания повышается до 98,8%. Такой режим используется, например, при вводе PIN-кода или телефонного номера, когда количество цифр в высказывании известно заранее.

Следует отметить, что алгоритмы системы никак не привязаны к конкретному языку и составу словаря. При наличии соответствующего речевого материала система может быть переобучена на любой другой набор слов и речевых команд, без каких либо затруднений.

В системе SPIRIT ASR Engine были реализованы как известные решения, такие как Скрытые Марковские модели (СММ), так и нестандартные подходы, позволившие значительно повысить надежность распознавания в реальных акустических условиях.

Использование СММ является на сегодняшний день наиболее популярным и успешно применяемым подходом к проблеме распознавания речи. В данной статье представлены основные принципы СММ.

В следующей статье будут рассмотрены алгоритмы распознавания слитно произнесенных цепочек слов, примененные в SPIRIT ASR Engine.

Скрытые Марковские модели

Рассмотрим основные принципы СММ. Более детальное обсуждение теории СММ и вопросов их применения в распознавании речи можно найти в [1,2].

Определение Скрытых Марковских моделей

Использование СММ для распознавания речи базируется на следующих предположениях:

1. Речь может быть разбита на сегменты (состояния), внутри которых речевой сигнал может рассматриваться как стационарный. Переход между этими состояниями осуществляется мгновенно.
2. Вероятность символа наблюдения, порождаемого моделью, зависит только от текущего состояния модели и не зависит от предыдущих порожденных символов.

По сути, ни одно из этих двух предположений не является справедливым для речевого сигнала. Боль-

шое количество исследований было посвящено тому, чтобы сгладить недостатки этих предположений [3]. Тем не менее, стандартные СММ являются основой для большинства современных систем распознавания речи.

Существует несколько типов СММ, различающихся по своей топологии (эргодические, лево-правые и др.), с дискретными или непрерывными символами наблюдения. Для построения данной ASR-engine использовались лево-правые СММ без пропусков состояний с непрерывной плотностью наблюдений, именно такие модели и будут рассмотрены. На *рис.1* представлена топология подобной СММ с тремя состояниями.

Скрытая Марковская модель представляет собой конечный автомат, изменяющий свое состояние в каждый дискретный момент времени *t*. Переход из состояния S_i в состояние S_j осуществляется случайным образом с вероятностью a_{ij}. В каждый дискретный момент времени модель порождает вектор o_t наблюдений с вероятностью $b_j(o_t)$.

Для определения СММ необходимо задать следующие элементы:

1. N – число состояний модели. В каждый момент времени модель может находиться в одном из N различных состояний $S_1, S_2,...,S_N$. В дискретные моменты времени t модель претерпевает изменение состояния (возможно, переходя при этом опять в то же состояние). Состояния модели в каждый момент времени будем обозначать как q_t.
2. Распределение вероятностей переходов между состояниями $A = \{a_{ij}\}$, где

$$a_{ij} = P[q_{t-1} = S_i \,|\, q_t = S_j],\ 1 \leqslant i, j \leqslant N. \qquad (1.1)$$

3. $B = \{b_j(o_t)\}$ – распределение плотностей вероятности наблюдений для каждого состояния S_j, где o_t – вектор наблюдений в момент времени t и

$$b_j(\boldsymbol{o}_t) = P(\boldsymbol{o}_t \,|\, q_t = S_j),\ 1 \leqslant t \leqslant T,\quad 1 \leqslant j \leqslant N \qquad (1.2)$$

В непрерывных СММ величина $b_j(o_t)$ моделируется конечной гауссовской смесью вида

$$b_j(\boldsymbol{o}_t) = \sum_{k=1}^{M} c_{jk}\, N(\boldsymbol{o}_t, \boldsymbol{\mu}_{jk}, U_{jk}), \qquad (1.3)$$

где c_{jk} – весовой коэффициент k-й компоненты смеси в состоянии j, M – количество компонент смеси и N – Гауссовская плотность вероятности. Коэффициенты c_{jk} удовлетворяют стохастическим ограничениям

$$\sum_{k=1}^{M} c_{jk} = 1, \qquad 1 \leqslant j \leqslant N \qquad (1.4)$$

$$c_{jk} \geqslant 0, \qquad 1 \leqslant j \leqslant N,\ 1 \leqslant k \leqslant M.$$

Плотность N характеризуется вектором средних значений $\boldsymbol{\mu}_{jk}$ и ковариационной матрицей U_{jk} для k-ой компоненты смеси в состоянии S_j:

$$N(\boldsymbol{o}_t, \boldsymbol{\mu}_{jk}, U_{jk}) =$$

$$= \frac{1}{\sqrt{(2\pi)^n |U_{jk}|}} \exp\left[-\frac{1}{2}(\boldsymbol{o}_t - \boldsymbol{\mu}_{jk})^T U^{-1}_{jk} (\boldsymbol{o}_t - \boldsymbol{\mu}_{jk})\right], \qquad (1.5)$$

где n – размерность вектора наблюдений $\boldsymbol{o}_t$.

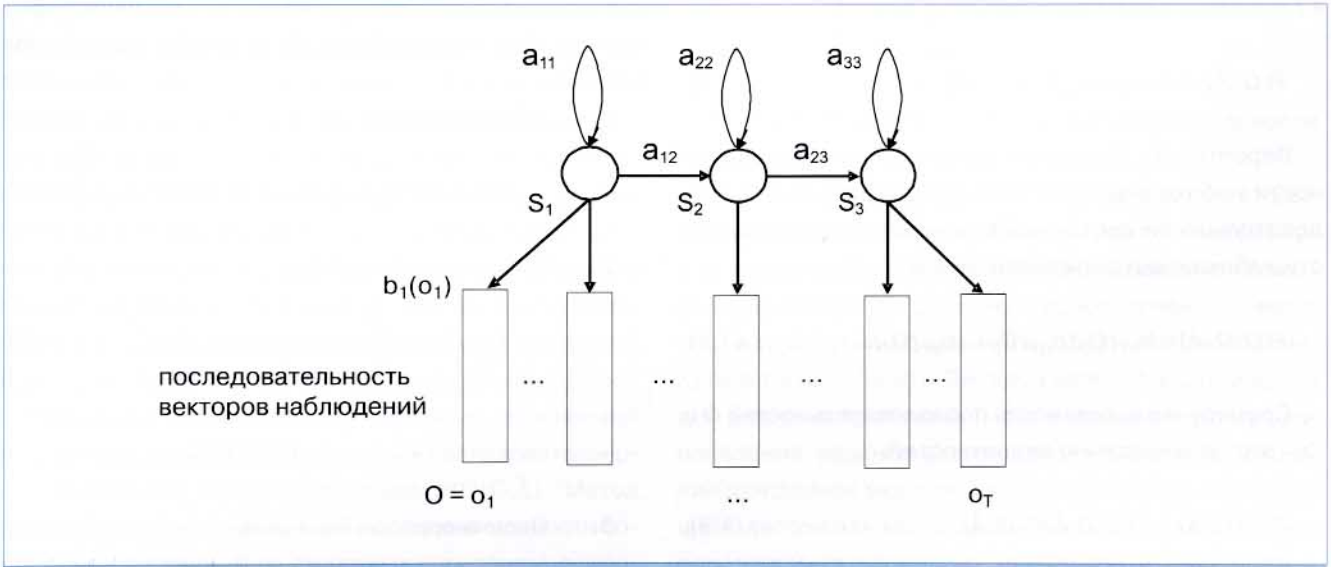

Рис. 1. Лево-правая СММ без пропусков состояний

4. Начальное распределение вероятностей состояний $\pi = \{\pi_i\}$

$$\pi_i = P[q_1 = S_i], \qquad 1 \leqslant i \leqslant N. \qquad (1.6)$$

Будем обозначать набор параметров, полностью определяющий СММ, как $\lambda = (A,B,\pi)$.

Существует три основные проблемы, которые необходимо решить, для того, чтобы модель, описанная выше, могла бы использоваться в практических задачах.

Проблема 1. Пусть заданы последовательность наблюдений $O = O_1,O_2,...,O_T$ и модель $\lambda = (A,B,\pi)$. Как эффективно вычислить вероятность $P(O|\lambda)$ появления этой последовательности наблюдений для данной модели?

Проблема 2. Пусть заданы последовательность наблюдений $O = O_1,O_2,...,O_T$ и модель $\lambda = (A,B,\pi)$. Как выбрать последовательность состояний $Q = q_1,q_2,...,q_T$, которая с наибольшей вероятностью для данной модели $P(O,Q|\lambda)$ порождает заданную последовательность наблюдений?

Проблема 3. Каким образом нужно подстроить параметры модели $\lambda = (A,B,\pi)$, для того, чтобы максимизировать $P(O|\lambda)$?

Далее последовательно будут рассмотрены эти три проблемы и алгоритмы, приводящие к их решению.

Алгоритм прямого-обратного хода (решение проблемы 1)

Наиболее прямой путь для вычисления вероятности $P(O|\lambda)$ – перечислить все возможные последовательности состояний заданной длины T и рассчитать соответствующие вероятности. Так, для фиксированной последовательности $Q = q_1,q_2,...,q_T$, вероятность её появления для данной модели λ равна

$$P(Q|\lambda) = \pi_{q1} a_{q1q2} a_{q2q3} \dots a_{qT\text{-}1qT}. \qquad (1.7)$$

Вероятность появления заданной последовательности наблюдений для этой фиксированной последовательности состояний при условии независимости наблюдений определяется как

$$P(O|Q,\lambda) = b_{q1}(O_1)\, b_{q2}(O_2)...b_{qT}(O_T). \qquad (1.8)$$

Совместная вероятность последовательностей *O* и *Q* – это произведение вероятностей:

$$P(O,Q|\lambda) = P(O|Q,\lambda)P(Q|\lambda). \qquad (1.9)$$

Вероятность появления последовательности наблюдений *O* для данной модели вычисляется как сумма всех этих совместных вероятностей для всех возможных последовательностей состояний *Q*:

$$P(O|\lambda) = \sum_Q P(O|Q,\lambda)\, P(Q|\lambda) =$$

$$= \sum_Q \pi_{q1} b_{q1}(O_1)\, a_{q1q2}\, b_{q2}(O_2)...a_{qT\text{-}1qT}\, b(O_T). \qquad (1.10)$$

Из последнего выражения следует, что при прямом подсчете вероятности $P(O|\lambda)$ требуется провести порядка $2TN^T$ вычислительных операций. Даже для небольших чисел N=10 и T=5 необходимо порядка 10^6 операций умножения. Ясно, что для практического решения проблемы 1 требуется более эффективная процедура. Такая процедура существует и называется процедурой прямого-обратного хода (the forward-backward procedure).

Существуют две модификации алгоритма, равноценные по вычислительным затратам – алгоритм прямого хода и алгоритм обратного хода. Эти алгоритмы различаются выбором ведущей переменной, прямой или обратной, которая предпочтительней в каждом конкретном случае.

Алгоритм прямого хода

Введем так называемую прямую переменную $\alpha_t(i)$, определяемую выражением

$$\alpha_t(i) = P(O_1,O_2,...,O_t,q_t = S_i|\lambda), \qquad (1.11)$$

которая представляет собой вероятность появления для данной модели частичной последовательности наблюдений $O_1,O_2,...,O_t$ до момента времени t и состояния S_i в этот момент времени. Значения переменной $\alpha_t(i)$ вычисляются по индукции следующим образом:

1. Инициализация:

$$\alpha_t(i) = \pi_i b_i(O_1), \; 1 \leqslant i \leqslant N. \qquad (1.12)$$

2. Индуктивный переход:

$$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i)\alpha_{ij}]\, b_j(O_{t+1}) \qquad (1.13)$$

$$1 \leqslant t \leqslant T-1,$$
$$1 \leqslant i \leqslant N$$

3. Окончание:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i). \qquad (1.14)$$

Для вычисления вероятности $P(O|\lambda)$ таким образом требуется уже порядка $N^T T$ [1] вычислительных операций. Для взятых в качестве примера чисел $N=10$ и $T=5$ это составляет около 500 операций умножения, что в 2000 раз меньше, чем для прямых вычислений.

Алгоритм обратного хода

Аналогичным образом можно ввести обратную переменную $\beta_t(i)$, определяемую выражением

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, \ldots, O_T \,|\, q_t = S_i, \lambda), \qquad (1.15)$$

которая для заданной модели λ представляет собой совместную вероятность появления частичной последовательности наблюдений от момента времени t+1 до T и состояния S_i в момент времени t. Значения обратной переменной также можно вычислить по индукции:

1. Инициализация:

$$\beta_T(i) = 1, \; 1 \leqslant i \leqslant N. \qquad (1.16)$$

2. Индукция:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j), \qquad (1.17)$$

для всех $t = T-1, T-2, \ldots, 1, \; 1 \leqslant i \leqslant N$.

3. Окончание:

$$P(O|\lambda) = \sum_{i=1}^{N} \pi_i b_i(O_1) \beta_1(i). \qquad (1.18)$$

Алгоритм Витерби (решение проблемы 2)

В отличие от проблемы 1, для которой можно дать точное решение, для проблемы 2, а именно поиска «оптимальной» последовательности состояний, вообще говоря, существует несколько возможных решений. Это обусловлено существованием нескольких приемлемых критериев оптимальности. Например, один из возможных критериев состоит в том, чтобы выбрать такие состояния q_t, каждое из которых, *взятое отдельно*, является наиболее вероятным. Можно также максимизировать вероятность появления пар состояний (q_t, q_{t+1}) или троек состояний (q_t, q_{t+1}, q_{t+2}) и т.д. Однако, наиболее широко используется критерий, основанный на поиске единственной наилучшей последовательности состояний, максимизирующий $P(Q|O,\lambda)$. Метод определения такой наилучшей последовательности состояний, основанный на динамическом программировании, называется алгоритмом Витерби.

Итак, для отыскания наилучшей последовательности состояний $Q = q_1, q_2, \ldots, q_T$ по заданной последовательности наблюдений $O_1, O_2, \ldots, O_T$ определим следующую величину:

$$\delta_t(i) = \max_{q_1 q_2 \ldots q_{t-1}} P[q_1, q_2 \ldots q_t = i, O_1, O_2 \ldots O_T | \lambda], \qquad (1.19)$$

которая представляет собой максимальную вероятность того, что при заданных t первых наблюдениях последовательность состояний заканчивается (в момент t) в состоянии S_i. Для того, чтобы затем восстановить последовательность состояний для всех значений t будем хранить значения аргументов, максимизирующих вероятность в массиве $\Psi_t(j)$. Полная процедура определения последовательности состояний будет выглядеть следующим образом:

1. Инициализация:

$$\delta_1(i) = \pi_i b_i(O_1), \; 1 \leqslant i \leqslant N; \qquad (1.20a)$$

$$\Psi_1(i) = 0. \qquad (1.20b)$$

2. Рекурсия:

$$\delta_t(j) = \max_{1 \leqslant i \leqslant N} [\delta_{t-1}(i) a_{ij}] b_j(O_t), \qquad (1.21a)$$
$$2 \leqslant t \leqslant T, \quad 1 \leqslant j \leqslant N;$$

$$\Psi_t(j) = \arg\max_{1 \leqslant i \leqslant N} [\delta_{t-1}(i) a_{ij}] \qquad (1.21b)$$
$$2 \leqslant t \leqslant T, \quad 1 \leqslant j \leqslant N.$$

3. Окончание:

$$P^* = \max_{1 \leqslant i \leqslant N} [\delta_T(i)], \qquad (1.22a)$$

$$q_T^* = \arg\max_{1 \leqslant i \leqslant N} [\delta_T(i)]. \qquad (1.22b)$$

4. Восстановление пути (последовательности состояний):

$$q_t^* = \Psi_{t+1}(q_{t+1}^*), \; t = T-1, T-2, \ldots, 1. \qquad (1.23)$$

Реализация алгоритма Витерби аналогична (за исключением шага восстановления) процедуре прямого хода. Основное отличие – использование вместо процедуры суммирования процедуры максимизации. Кроме того, алгоритм Витерби может быть применен для решения проблемы 1 (определения вероятности появления заданной последовательности наблюдений для данной модели), поскольку на окончательном шаге алгоритма мы получаем вероятность для всей предшествующей последовательности наблюдений (выражение 1.22a).

Алгоритм Баума-Уелша (решение проблемы 3)

Проблема переоценки параметров модели (A,B,π) по заданной последовательности наблюдений – наиболее трудоемкая (в вычислительном плане) проблема СММ. Не существует известного аналитического выражения для параметров такой модели. Тем не менее, используя итеративные процедуры, можно локально максимизировать вероятность $P(O|\lambda)$. Одна из таких процедур – метод переоценки Баума-Уелша (*Baum-Welch method*) или, что эквивалентно, EM-метод (*expectation-maximization*) .

Введем переменную:

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda), \qquad (1.24)$$

которая определяет вероятность того, что при заданной последовательности наблюдений в моменты времени t и t+1 система будет соответственно находиться в состояниях S_i и S_j.

Используя определения прямой и обратной переменных (выражения 1.11 и 1.15), можно записать:

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})b_{t+1}(j)}{P(O|\lambda)} = \qquad (1.25)$$

$$= \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})b_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(O_{t+1})b_{t+1}(j)}.$$

Введем также переменную $\gamma_t(i)$, являющуюся апостериорной вероятностью того, что при заданной последовательности наблюдений O система в момент времени t будет находиться в состоянии S_i:

$$\gamma_t(i) = \sum_{j=1}^{N}\xi_t(i,j). \qquad (1.26)$$

Если величину $\gamma_t(i)$ просуммировать по всем t, то результат можно рассматривать как ожидаемое время пребывания системы в состоянии S_i. Аналогичным образом результат суммирования $\xi_t(i,j)$ по всем t, можно рассматривать как ожидаемое число переходов из состояния S_i в S_j.

$\sum_{t=1}^{T-1}\gamma_t(i)$ – ожидаемое число (1.27a)

переходов из S_i.

$\sum_{t=1}^{T-1}\xi_t(i,j)$ – ожидаемое число (1.27b)

переходов из S_i в S_j.

Используя перечисленные выше формулы можно получить процедуру переоценки параметров СММ:

$$\bar{\pi}_i = \gamma_1(i), \qquad (1.28a)$$

$$\bar{a}_{ji} = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t}^{T-1}\gamma_t(i)}, \qquad (1.28b)$$

$$\bar{b}_j(o_t) = \sum_{k=1}^{M}\bar{c}_{jk}N(o_t, \bar{\mu}_{jk}, \bar{U}_{jk}). \qquad (1.28c)$$

Переоценка компонент $\bar{c}_{jk}$, $\bar{\mu}_{jk}$, $\bar{U}_{jk}$ в выражении 1.28c выполняется по следующим формулам:

$$\bar{c}_{jk} = \frac{\sum_{t=1}^{T}\gamma_t(j,k)}{\sum_{t}^{T}\sum_{k=1}^{M}\gamma_t(j,k)}, \qquad (1.29a)$$

$$\bar{\mu}_{jk} = \frac{\sum_{t=1}^{T}\gamma_t(j,k)\cdot o_t}{\sum_{t}^{T}\gamma_t(j,k)}, \qquad (1.29b)$$

$$\bar{U}_{jk} = \frac{\sum_{t=1}^{T}\gamma_t(j,k)\cdot(o_t-\mu_{jk})(o_t-\mu_{jk})^T}{\sum_{t}^{T}\gamma_t(j,k)}, \qquad (1.29c)$$

где $\gamma(j,k)$ – вероятность того, что при заданной последовательности наблюдений в момент времени t модель находится в состоянии j, причем наблюдаемый в это момент вектор o_t порожден k-й компонентой смеси, т.е.

(1.30)

$$\gamma_t(j,k) = \left[\frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N}\alpha_t(j)\beta_t(j)}\right]\left[\frac{c_{jk}N(o_t,\mu_{jk},U_{jk})}{\sum_{k=1}^{M}c_{jk}N(o_t,m_{jk},U_{jk})}\right].$$

Переоценка параметров модели λ по приведенным формулам приводит к возрастанию функции правдоподобия, то есть:

$$P(O|\bar{\lambda}) \geq P(O|\lambda). \qquad (1.31)$$

Распознавание речи на базе СММ

Существует множество различных способов использования СММ в целях распознавания речи. Выбор определенного метода определяется конкретной задачей, которую необходимо решить. Например, задачу распознавания изолированных слов (речевых команд) при словаре небольшого размера (порядка сотен

слов) можно эффективно решить, используя отдельную СММ для каждого слова. При таком подходе входная последовательность наблюдений пропускается через каждую СММ. Используя алгоритм прямого или обратного хода, или же алгоритм Витерби, вычисляется вероятность правдоподобия для каждого слова. В качестве результата выбирается слово, вероятность правдоподобия которого наибольшая, то есть :

$$V^* = \underset{1 \leqslant v \leqslant V}{arg\ max}(P(O|\lambda_v)), \qquad (1.32)$$

где V – размер словаря распознавания. Схематически этот процесс представлен на *рис. 2*.

Более сложной задачей является распознавание слитного высказывания, содержащего слова из заданного словаря. Одним из вариантов такой задачи является задача распознавания последовательности произносимых цифр. При этом базовой единицей остается слово, а слитная речь распознается как сочлененная последовательность из моделей слов. Задачи такого типа называются задачами распознавания связанных (конкатенированных) слов. Методы ее решения будут приведены в следующем разделе.

Алгоритмы распознавания слитно произнесенных речевых команд, применяемые в SPIRIT ASR Engine

В данном разделе описаны основные алгоритмы, использованные при построении системы распознавания слитно произнесенных речевых команд.

Перед тем как перейти к рассмотрению собственно алгоритмов, мы коснемся некоторых аспектов, связанных с корректным моделированием длительности состояний СММ.

Моделирование длительности состояний

Можно показать (см. [1]), что плотность вероятности $p_j(d)$ пребывания в состоянии S_j, с переходной вероятностью a_{jj} имеет вид:

$$p_j(d) = (a_{jj})^{d-1}(1 - a_{jj}). \qquad (2.1)$$

Однако для большинства физических сигналов эта плотность вероятности для длительности пребывания в состоянии является неприемлемой. Более предпочтительным вариантом является моделирование длительности состояния явным образом. На *рис. 3* в качестве примера такого моделирования представлена гистограмма вероятности $p_j(d)$ для первого состояния слова «six», соответствующего начальному звуку /s/. Гистограмма получена эвристически на основе разметки на состояния. Отметим, что подобные распределения хорошо аппроксимируются Гамма функцией (см. *рис. 3*).

Введение явным образом информации о длительности, накладывает дополнительное ограничение на длительность пребывания модели в одном состоянии $d_{min} \leqslant d \leqslant d_{max}$. При наложении такого ограничения использование информации только от предыдущего состояния приводит к некорректной работе алгоритма Витерби. Это будет ясно из следующего примера.

Рассмотрим участок декодирования, представленный на *рис. 4*. Цифрами условно обозначены вероятности появления символов наблюдения $b_j(o_t)$.

Для состояния S_j в момент времени t, следуя правилу максимизации, мы должны выбрать состояние S_{j-1}, как наиболее вероятное предыдущее состояние. Предположим, что на длительность состояния S_j наложено ограничение $d_j \geqslant 2$. Тогда единственно возможным переходом из состояния S_j в момент времени *t+1* будет переход в то же состояние. Вследствие этого к точке к моменту *t+2* бу-

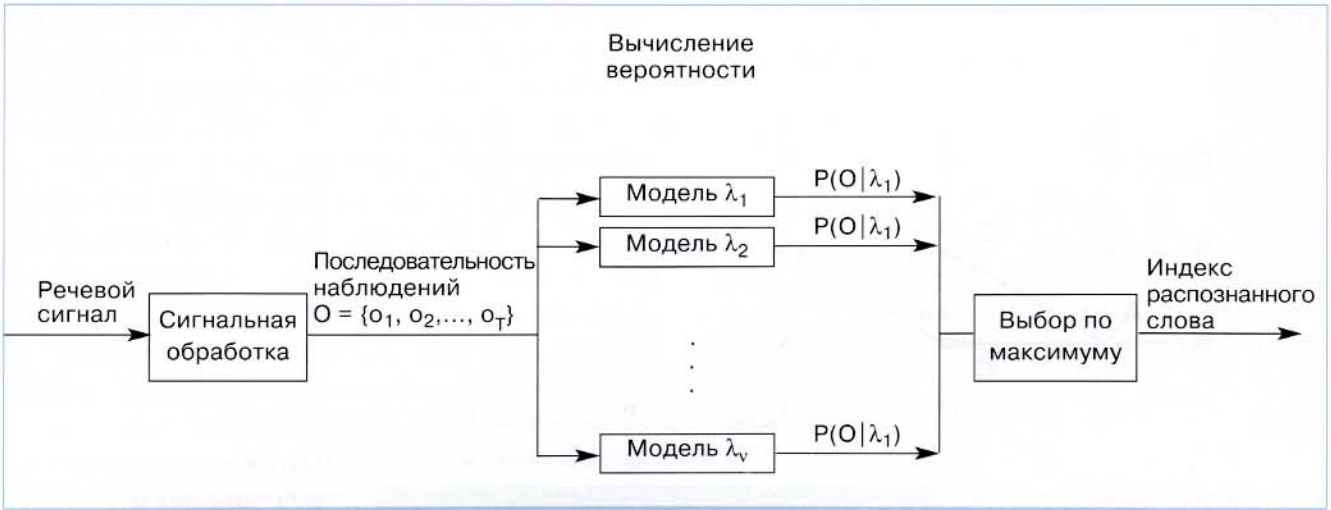

Рис. 2. Блок-схема распознавателя изолированных слов на основе СММ

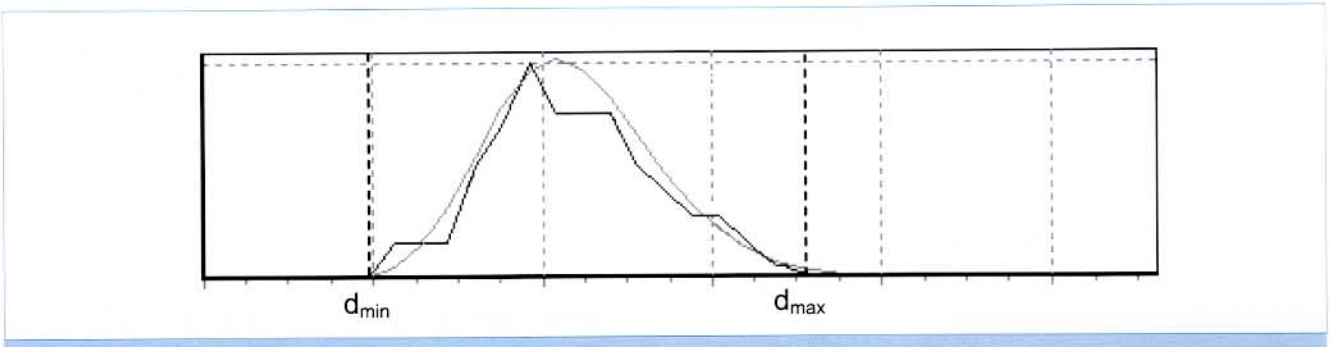

Рис. 3. Гистограмма плотности распределения длительности первого состояния слова «six»

дет пройдена траектория *ACDFG* и накоплена вероятность p = 0,6·0,6·0,6·0,1·0,6 = 0,01296. Тогда как наиболее оптимальной траекторией в этом случае будет траектория *ABDEG* с вероятностью p = 0,6·0,3·0,6·0,6·0,6 = 0,03888. Для устранения этой проблемы был разработан «модифицированный» алгоритм Витерби .

Модифицированный алгоритм Витерби

В предлагаемой модификации алгоритма Витерби мы будем максимизировать не предыдущее состояние, а длительность текущего. Для этого определим величину $\varphi_j^t(d)$, которая представляет собой вероятность того, что состояние *j* окончилось в момент времени t и началось в момент времени t-d, т.е. имело продолжительность *d*:

$$\varphi_j^t(d) = \begin{cases} \prod_{t'=t-d}^{t} b_j(O_{t'}) \cdot (p_j(d))^{\alpha} \cdot \delta_{j-1}(t-d), & 1 < j \leqslant N, \\ \prod_{t'=t-d}^{t} b_j(O_{t'}) \cdot (p_j(d))^{\alpha}, & j = 1, \end{cases} \quad (2.2)$$

где

$p_j(d)$ – плотность вероятности длительности состояния,

α – масштабный коэффициент для длительностей состояний,

$\delta_{j-1}(t-d)$ – вероятность, накопленная к моменту *t-d*.

Введем так же дополнительную переменную $D_j(t)$, обозначающую наиболее вероятную длительность состояния S_j, при условии его окончания в момент времени *t*.

Полную процедуру, требуемую для определения последовательности состояний, можно сформулировать следующим образом:

1) Инициализация:

Для всех состояний j = 1...N:

$$\delta_j(1) = \begin{cases} b_j(O_1), & j = 1, \\ 0, & j > 1. \end{cases} \quad (2.3a)$$

$$D_j(1) = 0. \quad (2.3b)$$

2) Рекурсия:

Для всех состояний j = 1...N:

если j > 1:

$$dj(t) = \max_{t_s \leqslant d \leqslant t_e} \varphi_j^t(d), \quad (2.4a)$$

где

$t_s = max(1, t - d_{j\,max})$, $t_e = max(1, t - d_{j\,min})$;

$$D_j(t) = \arg\max_{t_s \leqslant d \leqslant t_e} \varphi_j^t(d); \quad (2.4b)$$

если j = 1:

$$\delta_j(t) = \varphi_j^t(t), \quad (2.4c)$$

$$D_j(t) = t \quad (2.4d)$$

3) Окончание:

Итоговая вероятность:

$$P(O) = \max_j \delta_j(T). \quad (2.5a)$$

Последнее состояние:

$$q(T) = \arg\max_j \delta_j(T). \quad (2.5b)$$

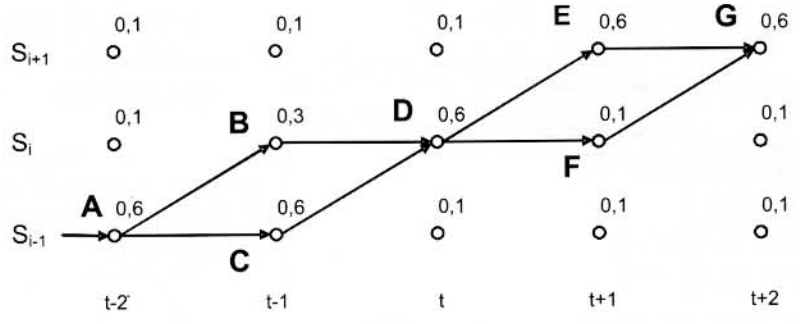

Рис. 4. Пример ошибочного декорирования вследствие ограничения на длительность состояния

4) Восстановление пути:

$$t = T, \quad (2.6a)$$
$$q = q(t), \quad (2.6b)$$
$$d = D_q(t), \quad (2.6c)$$

Пока $d \neq 0$

$$q_{t-d} \ldots q_t = q, \quad (2.6d)$$
$$q = q - 1, \quad (2.6e)$$
$$d = D_q(t). \quad (2.6f)$$

Шаг 1 устанавливает значение вероятности для первого состояния равной вероятности наблюдения символа O_1. Для остальных состояний вероятности устанавливаются в ноль. Индуктивный переход, в котором заключается основное содержание алгоритма, проиллюстрирован на *рис. 5*. Вероятность окончания состояния в данной точке складывается из совместной вероятности наблюдения символов $O_{t-d},\ldots,O_t$ на интервале {t – d,t}, вероятности длительности состояния p(d) и вероятности окончания предыдущего состояния в точке t – d.

На заключительном этапе производится максимизация по наиболее вероятному последнему состоянию и от него восстанавливается вся последовательность состояний.

Далее в тексте, если это не оговорено особо, под алгоритмом Витерби будет пониматься именно модифицированный вариант.

Алгоритм Витерби для цепочки связанных слов

Поиск оптимальной последовательности состояний для цепочек связанных слов производится на всем пространстве состояний моделей. Помимо моделей слов, для повышения надежности распознавания и устранения неточностей определения граничных точек, добавляется модель паузы. Для модели паузы из вычислений исключаются плотности вероятностей длительности состояний *p(d)*. Таким образом из моделей отдельных слов формируется одна связанная СММ для всех слов, включая паузу (*рис. 6*).

Для организации переходов между словами вводится дополнительная переменная – вероятность конца слова в данной точке $P_w(t)$, равная вероятности конца последнего состояния в этой же точке. Переход в новое слово происходит из наиболее вероятного предыдущего слова. Для того, чтобы затем восстановить последовательность состояний, будем использовать массив $\psi_t(j)$, в котором сохраняются индекс состояния предшествующего данному. Кроме того, в массиве $Pt_w(t)$ отдельно сохранятся индекс последнего состояния наиболее вероятного слова, завершившегося в точке t.

Для того, чтобы обеспечить связь моделей при переходе в новое слово, необходимо изменить процедуру вычисления величины $\varphi_j^t(d)$ для первого состояния следующим образом:

$$\varphi_j^t(d) = \prod_{t'=t-d}^{t} b_j(O_{t'}) \cdot (p_j(d))^{\alpha} \cdot P_w(t-d), \quad j = 1. \quad (2.7)$$

Теперь алгоритм Витерби для связанных цепочек будет выглядеть следующим образом:

1) Инициализация:

base = 0 – индекс первого состояния слова,

Для всех слов W_k:

Для всех состояний $j = 1 \ldots N_k$:

$$\delta_{j+base}(1) = \begin{cases} b_{j+base}(O_1), & j = 1 \\ 0, & j > 1 \end{cases}, \quad (2.8a)$$

$$D_{j+base}(1) = 0. \quad (2.8b)$$

$$base = base + N_k, \quad (2.8c)$$

2) Рекурсия:

Вычисление вероятностей по состояниям:

base = 0,

Для всех слов W_k:

Для всех состояний $j = 1 \ldots N_k$:

если $j > 1$:

$$\delta_{j+base}(t) = \max_{t_s \leqslant d \leqslant t_e} \varphi_{j+base}^t(d), \quad (2.9a)$$

$$D_{j+base}(t) = \arg\max_{t_s \leqslant d \leqslant t_e} \varphi_{j+base}^t(d), \quad (2.9b)$$

$$\psi_t(j + base) = j + base - 1, \quad (2.9c)$$

если $j = 1$:

$$\delta_{j+base}(t) = \max_{t_s \leqslant d \leqslant t_e} \varphi_{j+base}^t(d), \quad (2.9d)$$

$$D_{j+base}(t) = \arg\max_{t_s \leqslant d \leqslant t_e} \varphi_{j+base}^t(d), \quad (2.9e)$$

$$\psi_t(j + base) = St_w(t - D_{j+base}(t)), \quad (2.9f)$$

$$base = base + N_k, \quad (2.9g)$$

Сохранение вероятности для наиболее вероятного слова W_{best}, окончившегося в данной точке:

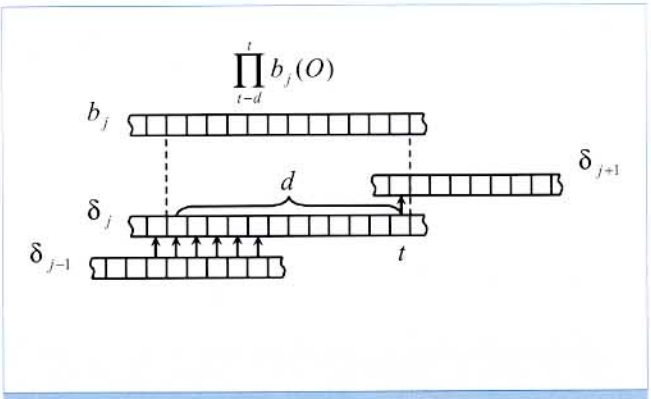

Рис. 5. Пример последовательности операций, необходимых для вычисления вероятности окончания состояния $\varphi_j^t(d)$ в момент времени t, при условии его начала в момент t-d

$base = 0,$

$P_{best} = 0,$

Для всех слов W_k:

если $\delta_{base+N_k}(t) > P_{best}$

$$P_{best} = \delta_{base+N_k}(t), \quad (2.10a)$$

$$St_{best} = base + N_k, \quad (2.10b)$$

$$base = base + N_k; \quad (2.10c)$$

$$P_w(t) = P_{best}, \quad (2.10d)$$

$$St_w(t) = St_{best}. \quad (2.10e)$$

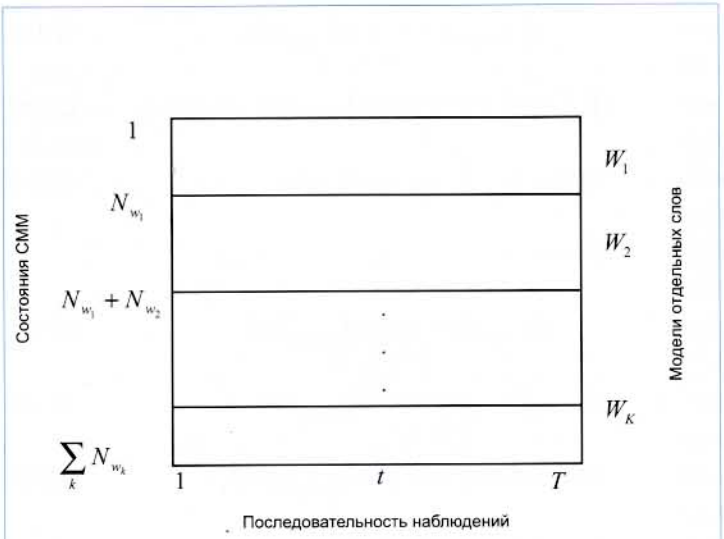

Рис. 6. Объединенная СММ для цепочки связанных слов

3) Окончание:

Итоговая вероятность:

$$P(O) = \max_j \delta_j(T). \quad (2.11a)$$

Последнее состояние:

$$q(T) = \arg\max_j \delta_j(T). \quad (2.11b)$$

4) Восстановление пути:

$$t = T, \quad (2.12a)$$

$$q = q(t), \quad (2.12b)$$

$$d = D_q(t), \quad (2.12c)$$

Пока $d \neq 0$

$$q_{t-d} \cdots q_t = q, \quad (2.12d)$$

$$t = t - d, \quad (2.12e)$$

$$d = D_q(t), \quad (2.12f)$$

$$q = \psi_t(q). \quad (2.12g)$$

Из восстановленной последовательности состояний путем несложных вычислений восстанавливается последовательность слов.

Вычислительные аспекты применения алгоритма Витерби

Поскольку для вычисления вероятности продолжительности состояния φ используется произведение вероятностей наблюдения символа $b(O_t)$, каждая из которых, как правило, существенно меньше единицы, то с ростом t вероятность φ начинает экспоненциально стремиться к нулю. Для устранения возможного переполнения необходимо либо вводить процедуру масштабирования, либо переходить к логарифмическим значениям вероятностей. Второй вариант более предпочтителен, поскольку позволяет для вычисления условных вероятностей заменить операцию умножение на сложение.

В логарифмическом виде формула вычисления вероятности длительности состояния будет иметь следующий вид :

$$\varphi_j^t(d) = \sum_{t'=t-d}^{t} \log(b_j(O_{t'})) + \alpha \cdot \log(p_j(d)) + \delta_j(t-d). \quad (2.13)$$

Логарифм априорной вероятности длительности состояния $p_j(d)$ может быть рассчитан заранее на этапе подготовки моделей. Логарифмические

величины вероятностей наблюдения символа $b(O_t)$ удобно вычислить предварительно и сохранить в массиве следующим образом:

$$b^{log}(1) = log(b(O_1)), \quad \text{для } t = 1, \tag{2.14a}$$

и далее рекуррентно

$$b^{log}(t+1) = b^{log}(t) + log\,(b(O_{t+1})). \tag{2.14b}$$

В таком случае вычисление φ становится еще более удобным:

$$\varphi_j^t(d) = b^{log}(t) - b^{log}(t-d) + \alpha \cdot p_j^{log}(d) + \delta_j(t-d). \tag{2.15}$$

Алгоритм построения уровней

В рассмотренном выше методе предполагается декодирование цепочки команд произвольной длины. Достоинством этого метода является относительная простота реализации и универсальность применения.

Рассмотрим возможные типы ошибок, возникающие при таком подходе.

1. Ошибка типа замена (одна команда заменяется другой).
2. В случае быстрого слитного произнесения команд возможно объединение двух команд в одну (ошибка типа удаление).
3. При медленном произнесении возможно разделение команды на две (ошибка типа вставка).
4. В условиях сильных нестационарных шумов возможны вставки на месте паузы в районе высокой энергии речеподобного шума.

Существует широкий класс приложений, где можно существенно уменьшить количество ошибок типа 2 – 4. Это возможно в случае, когда заранее известна некая информация о количестве слов в строке. Длина строки может быть жестко фиксирована, (например, в случае голосового ввода PIN-кода), иметь верхние и нижние ограничения или некий фиксированный набор разрешенных значений (например, в случае голосового ввода телефонного номера).

Как можно корректно использовать эту информацию? Можно заметить, что в рассматриваемом выше подходе узлами промежуточной максимизации вероятностей, являются точки перехода из команд в паузу или следующую команду. Таким образом при прямом проходе распознавания ошибки типа 2 – 4 неустранимо вносятся в путь декодирования. Для исправления этой ситуации предлагается разделить узлы максимизации для каждого уровня.

Под уровнем L здесь и далее будем понимать состояние системы, содержащее L завершенных команд, в момент от начала сигнала до текущего момента времени t. Вероятность нахождения системы в момент времени t на уровне L обозначим как $P_L(t)$.

В данном подходе целесообразно рассматривать модель для каждой команды как независимый «черный ящик», на вход которого в каждый момент времени t подается вектор речевых параметров, а на выходе – массив абсолютных вероятностей $P_w(d)$, соответствующих всем возможным длительностям d команды w. $P_w(d)$ вычисляется стандартным (пункт 1.3) или модифицированным (пункт 2.2) алгоритмом Витерби при условии, что вероятность входа в первое состояние модели не зависит от текущего состояния системы и всегда равна 1. Все модели команд просчитываются в режиме реального времени независимо. При этом не происходит дублирования процедуры декодирования команды для каждого уровня. В общем случае, внутренняя структура моделей значения не имеет и запоминание путей декодирования внутри модели может не вестись.

Введем вероятность выхода из модели w в момент времени t на уровне L:

$$P_{wL}(t) = \max_d (P_w(d) \cdot P_{L-1}(t-d)). \tag{2.16}$$

Тогда вероятность нахождения системы в момент времени t на уровне L будет вычисляться как:

$$P_L(t) = \max_w \left[\max(P_{wL}(t)), P_L(t-1) \cdot B_{pause}(t) \right], \tag{2.17}$$

где $B_{pause}(t)$ – вероятность паузы в момент времени t, а член $P_L(t-1) \cdot B_{pause}(t)$ соответствует вероятности наблюдения паузы на уровне L (уровень при переходе в паузу не меняется). Здесь для упрощения не учитываются (полагаются равными 1) переходные вероятности на уровне команд. В реальном применении учет этих вероятностей также можно опустить. Схема вычисления нового значения $P_L(t)$ для одного уровня представлена на *рис. 7*.

Двигаясь по шкале времени, мы получаем значения $P_L(t)$ для всех уровней от 0 до максимального значения L_{max}. При этом мы можем определить наиболее вероятное количество команд в строке как

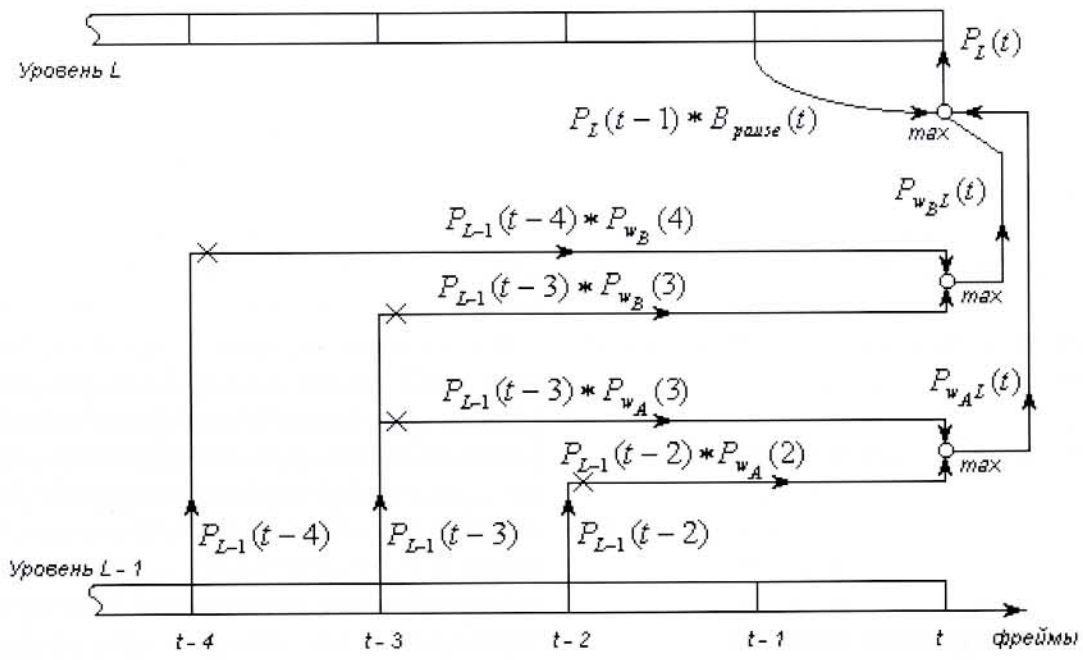

Рис. 7. Пример вычисления вероятностей по схеме построения уровней. Здесь предполагается, что существуют только 2 команды: w_A и w_B с разрешенными длительностями 2..3 и 3..4 соответственно

$$L_{best}(t) = \arg\max_{L} P_L(t) \qquad (3.18)$$

В случае режима реального времени ввод сигнала и процесс распознавания прекращаются при выполнении условия равенства $L_{best}(t)$ желаемому количеству команд. В случае мягких ограничений на длину строки необходимо использовать более сложные алгоритмы остановки процесса распознавания, рассмотрение которых выходит за рамки этой статьи.

После остановки процесса распознавания (при выполнении указанного выше условия или по достижению конца файла) на этапе обратного декодирования все уровни, кроме желаемого, игнорируются.

Следует отметить возможность использования в данном подходе маски ввода, т.е. когда часть команд известна. В частности, этот алгоритм использовался для автоматической разметки в большей части тренировочной базы данных.

Заключение

В работе рассмотрены основные принципы распознавания речи на базе Скрытых Марковских моделей и методы их использования при построении SPIRIT ASR-engine. Описаны основные алгоритмы, применяемые в разработанной системе распознавания речевых команд и связной речи, которые используют как классические методы, так и нестандартные подходы при декодировании и оптимизации.

Результаты работы SPIRIT ASR-engine показывают на примере распознавыания английских цифр высокую эффективность изложенных подходов в решении задач распознавания изолированных и слитно произнесенных команд в дикторонезависимом режиме.

В следующей статье мы планируем затронуть проблему устойчивой работы систем распознавания в условиях зашумленной акустической обстановки. Будут рассмотрены методы и практические результаты решения этой проблемы.

Литература

1. *Л.Р. Рабинер*, «Скрытые марковские модели и их применение в избранных приложениях при распознавании речи», ТИИЭР, т. 77, №2, февраль 1989.

2. *L.R. Rabiner, B-H Young*. Fundamentals of the speech recognition. Prentice Hall, Englewood Cliffs, NJ, 1993.

3. *M. Gales*, «The Theory of Segmental Hidden Markov Models», Cambridge University, 1993.