TR－H－067

最小キーワード列分類誤りのための新しいスポッタ設計法

小森 隆
片桐 滋

1994. 3. 28

ATR人間情報通信研究所
〒619-02 京都府相楽郡精華町光台2-2　☎07749-5-1011

ATR Human Information Processing Research Laboratories
2-2, Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02 Japan
Telephone: +81-7749-5-1011
Facsimile: +81-7749-5-1008

最小キーワード列分類誤りのための新しいスポッタ設計法
A Novel Spotter Design Method for Minimum Error Classification of Keyword-Sequences

小森 隆 *　片桐 滋 **

*Takashi Komori * and Shigeru Katagiri ***

* ATR 人間情報通信研究所

* ATR Human Information Processing Research Laboratories

** ATR 音声翻訳通信研究所

** ATR Interpreting Telecommunications Research Laboratories

概要

本技術覚書は次の 2 部からなる.

1. スポッティング型連続音声認識器の誤り率および計算量の観点からの最適化法
2. A Novel Spotting-Based Approach to Continuous Speech Recognition: Minimum Error Classification of Keyword-Sequences

第 1 部では，組合せ決定問題推論機構の定式化とその最適化設計法について述べ，その設計法がスポッティング型連続音声認識器にも応用可能であることを示している．第 2 部 (英語) では，実際のスポッティング型連続音声認識器に本設計法を適用し，日本語連続文中からのキーワード抽出実験を行った結果も記している.

スポッティング型連続音声認識器の誤り率および計算量の観点からの最適化法

小森 隆
(株)ATR 人間情報通信研究所

概要

単語スポッティングに基づく連続音声認識器は，概ね，スコアに基づく部分仮説の絞り込みと，部分仮説の組合せによる単語列の決定という二つの処理機構から構成される．従来，その設計法は，構成上の複雑さのため，部分的にのみ最適なものや経験的なものにとどまっており，理論的な背景を持つ系全体の最適化法は提案されていなかった．そこで，本稿では，単語スポッティングに基づく連続音声認識器を，仮説の対数事後オッズの累積に基づく組合せ決定機構として定式化し，最小決定誤り規準に基づく設計法を提案する．さらに，予備選択処理も含めた枠組での，計算量の節約のための最適化法も提案する．これらの設計法は音声認識のみならず推論機構一般に適用できる．

1 はじめに

人間による自然な発話は，必ずしも文法規則に従うとは限らないうえ，間投詞や舌打ちなど様々な音響現象が混入している．これを機械への入力インタフェースとして用いるための有望な方法の一つは，発話音声からシステムにとって重要な限定された単語のみをその位置とともに検出し(ワードスポッティング)，その結果をもとに構文解析などの高次処理を経て最も確からしい単語列を決定するものである．

ワードスポッティングの一つの方法は，対象単語との一致性を測る何らかのスコアを観測音声の時刻ごとに計算し，しきい値との比較によって単語候補を決定するものである [1, 2, 3, 4]．ただしこの場合，しきい値は経験的に選ばれていた．さらに，個々の単語候補のスコアは，しきい値と比較するために単語の時間長に依存しないように正規化されるため，長さの異なる単語列の確からしさを単語スコアの単純な累積によって直接比較することはできない．主にこの理由のため，この枠組における従来の単語列スコアも経験則に基づくもののみであった [1, 2, 4]．

ワードスポッティングのもう一つの方法は，対象単語の各類の他に，「ごみ類」，すなわち，対象単語以外の音響現象をすべて包含する類，を用意し，観測音声をそれらの類の音響現象が隙間なく連続したものと見なして単語列を認識するものである [5]．この方法は，従来の連続音声認識の枠組と全く同じ問題に煩わされる．すなわち，各時刻において累積スコアを文法ノードの数だけ保存かつ計算しなければならないが，文法の複雑さの増加に伴い，文法ノード数は爆発的に増大し，計算は非現実的なものになる．これを防ぐため，累積スコアの低い文法ノードを途中で棄却する方法(ビームサーチ)が取られる．この方法において単語候補レベルの絞り込みを行うことは簡単ではない．

両者に共通の戦略は，観測音声に対して最も確からしい単語列を求めるための膨大な計算量を，理論的最適性を犠牲にし，経験的知識に基づく候補絞り込みによって削減することである．この場合，たとえ二つの部分を最適に設計しても，それぞれに異なる規準を用いている限り，システム全体を最適化することにはならない．スコア計算部と候補絞り込み部の両方を含んだ系全体を単一の目的関数によって直接的に最適化すべきである．ところが，ワードスポッ

ティングに基づく音声認識器の従来の設計法においては，スコア計算部の設計と候補絞り込み処理部の設計は別々になされていた．

近年発表された一般化確率的降下法 (GPD)[6] は，一次微分可能な連続関数による近似の採用により，最小値選択や決定処理という不連続な処理を含む系の勾配探索による最適化の枠組を提供した．その具体的応用として，音声の分類における誤識別率を最小化する学習法が提案されている [7] ほか，スポッティングにも応用されている [8, 9]．

本稿では，ワードスポッティングによる連続単語認識器を GPD に基づいて最適化する設計法を検討する．ワードスポッティングによる連続単語認識は個々の単語候補の決定の組合せであると考えられるので，第一段階として，一般的な組合せ決定問題を解く機構の定式化をする．第二段階として，その最適化設計法を検討する．設計規準は，最尤推定，最小組合せ決定誤り，最小組合せ決定誤りに最小計算量を加味したもの，の 3 種類を検討する．第三段階で，具体的な音声認識器への実装例を示す．

2 組合せ決定機構の定式化

2.1 対数事後オッズに基づく決定機構

証拠 x を得て結論 a をくだす決定機構について考える．選びうる結論の集合を $\mathcal{A} = \{\alpha_j\}_{j=1}^{J}$ とすると，一般には，各々の α_j が互いに排反であるとは限らないので複数の結論が導かれることもあり得，また，証拠 x からはいずれの結論も導き出せない，ということも起こりうる．このような場合は $a \in \mathcal{A}$ ではなく，$a \in \mathcal{P} \subseteq \wp(\mathcal{A})$ とするべきである．ここに，$\wp(\mathcal{A})$ は $\mathcal{A}$ のべき集合であり，$|a| > 1$ の場合は複数の結論が導かれることを，$a = \emptyset$ の場合はいずれの結論も導かれないことを表す．部分的に排反な結論もありうるので，一般に $\mathcal{P}$ は $\wp(\mathcal{A})$ の部分集合である．証拠 x を得たときの結論 a の事後確率 $\Pr(a \mid x)$ が与えられるとき，Bayes の決定則，すなわち

$$a^* = \arg\max_{a \in \mathcal{P}} \Pr(a \mid x) \tag{1}$$

なる a^* を選ぶことが決定誤りを最小にする．

いま，証拠 x を得たときに各々の結論 α_j が導き出される事後確率 $\Pr(\alpha_j \mid x)$ が独立であると仮定すると，

$$\Pr(a \mid x) = \prod_{j=1}^{J} \Pr(\alpha_j \mid x)^{1(\alpha_j \in a)} \{1 - \Pr(\alpha_j \mid x)\}^{1(\alpha_j \notin a)} \tag{2}$$

が成り立つ．ただし，$1(\cdot)$ はカッコ内の論理式が真のとき 1，偽のとき 0 をとる二値関数である．ここで，a^* は，結論 a の結論 $\emptyset$ に対する事後確率の比の対数によるスコア

$$S_a(x) = \ln \frac{\Pr(a \mid x)}{\Pr(\emptyset \mid x)} \tag{3}$$

$$= \sum_{j=1}^{J} 1(\alpha_j \in a) \ln \frac{\Pr(\alpha_j \mid x)}{1 - \Pr(\alpha_j \mid x)} \tag{4}$$

を用いて，

$$a^* = \arg\max_{a \in \mathcal{P}} S_a(x) \tag{5}$$

によっても得られる．式 (4) において，

$$O(\alpha_j \mid x) = \frac{\Pr(\alpha_j \mid x)}{1 - \Pr(\alpha_j \mid x)} \tag{6}$$

は証拠 x に基づく結論 α_j の確からしさを表す量で，事後オッズと呼ばれる．式 (5) は，結論 a の確からしさが個々の結論 $\alpha_j \in a$ の対数事後オッズの和によって見積もられ，その最大値の探

索によって最良の結論 a^* を導き出せることを示している．この考えに基づき，以下では，対数事後オッズの和によるスコア $S_a(x)$ の推定値の比較により決定を下す決定機構を取り扱う．すなわち，対数事後オッズ $\ln O(\alpha_j \mid x)$ の推定値が決定機構の系全体のパラメータ集合 Λ によって $\eta_j(x\,;\Lambda)$ として得られるとき，スコア $S_a(x)$ の推定値

$$\hat{S}_a(x\,;\Lambda) = \sum_{j=1}^{J} 1(\alpha_j \in a)\eta_j(x\,;\Lambda) \tag{7}$$

の最大値の探索によって決定を下すものである．

2.2 対数事後オッズの累積スコアによる組合せ決定機構

次に，複数の決定問題の組合せを解く場合について考える．I 個の問題の組 $Q = \{q_i\}_{i=1}^{I}$ の各要素 q_i についての結論をそれぞれ $a_i \in \mathcal{P}$ とする．それぞれに対する証拠 x_i の組 $X = \{x_i\}_{i=1}^{I}$ が得られたとき，選びうる結論の組の集合 $\Omega \subseteq \mathcal{P}^I$ のなかで最も確からしい結論の組を求めることを考える．各問題 q_i が互いに独立であるとき，結論の組 $A = \{a_i\}_{i=1}^{I} \in \Omega$ の事後確率について

$$\Pr(A \mid X) = \prod_{i=1}^{I} \Pr(a_i \mid x_i) \tag{8}$$

が成り立つので，2.1節と同様に結論の組 A のスコアを

$$S_A(X) = \ln \frac{\Pr(A \mid X)}{\prod_{i=1}^{I} \Pr(\emptyset \mid x_i)} \tag{9}$$

$$= \sum_{i=1}^{I}\sum_{j=1}^{J} 1(\alpha_j \in a_i) \ln O(\alpha_j \mid x_i) \tag{10}$$

と定めれば，

$$A^* = \arg\max_{A \in \Omega} S_A(X) \tag{11}$$

なる結論の組 A^* を選ぶことは Bayes の決定則と等価である．実際には真の事後確率 $\Pr(A \mid X)$ は未知なので，対数事後オッズの推定値の和で定義されるスコア

$$\hat{S}_A(X\,;\Lambda) = \sum_{i=1}^{I}\sum_{j=1}^{J} 1(\alpha_j \in a_i)\eta_j(x_i\,;\Lambda) \tag{12}$$

を最大化するもの

$$A^* = \arg\max_{A \in \Omega} \hat{S}_A(X\,;\Lambda) \tag{13}$$

を最適な結論の組と見なす．

2.3 対数事後オッズの累積スコアによる組合せ分類機構

次に，証拠の組 X によって結論の組 A の全体集合 Ω の互いに排反な C 個の類 $\Omega_1, \Omega_2, \ldots, \Omega_C$ のいずれかに分類する場合を考える．分類の場合は，Bayes の決定則は

$$c^* = \arg\max_{c} \Pr(\Omega_c \mid X) \tag{14}$$

$$= \arg\max_{c} \sum_{A \in \Omega_c} \Pr(A \mid X) \tag{15}$$

なる c^* を分類結果として採用することである．この決定則は，これと等価な次の決定則

$$c^* = \arg\max_c \left(\ln \sum_{A \in \Omega_c} \frac{\Pr(A \mid X)}{\prod_{i=1}^{I} \Pr(\emptyset \mid x_i)} \right) \tag{16}$$

$$= \arg\max_c \left(\ln \sum_{A \in \Omega_c} \exp(S_A(X;\Lambda)) \right) \tag{17}$$

で置き換えてもよい．実際には，真の $S_A(X;\Lambda)$ の値は未知なので，その推定値 $\hat{S}_A(X;\Lambda)$ によって類ごとのスコアを

$$\hat{S}_c^{sum}(X;\Lambda) = \ln \sum_{A \in \Omega_c} \exp(\hat{S}_A(X;\Lambda)) \tag{18}$$

とし，

$$c^* = \arg\max_c \hat{S}_c(X;\Lambda) \tag{19}$$

なる c^* を分類結果とすればよい．ただし，すべての類に含まれるすべての元 A についてスコア $S_A(X)$ を求めることは現実的には難しいので，

$$\hat{S}_c^{max}(X;\Lambda) = \exp\left(\max_{A \in \Omega_c} \hat{S}_A(X;\Lambda) \right) \tag{20}$$

で代用することもある．最大値の探索には動的計画法を用いることができて計算量がずっと少なくて済むからである．以降では，両者を一般化したスコア

$$\hat{S}_c(X;\Lambda) = \frac{1}{\xi_C} \ln \sum_{A \in \Omega_c} \exp(\xi_C \hat{S}_A(X;\Lambda)) \tag{21}$$

を用いる．ただし，ξ_C は正の定数である．$\hat{S}_c(X;\Lambda)$ は，$\xi_C = 1$ のとき式 (18) に一致し，$\xi_C \to \infty$ のとき式 (18) に限りなく近付く．

2.4 組合せ決定・分類のための候補絞り込み

2.2節のような組合せ決定や 2.3節のような組合せ分類を計算機の上で実現するためには，考えられるすべての結論 A についてスコア $\hat{S}_A(X;\Lambda)$ を求める必要がある．ところが結論の組合せ A の総数は $O(2^{JI})$ であるため，I や J の数の大きさに伴い爆発的に増大する．最大値探索においては動的計画法を用いればある程度減らせるが，それよりさらに計算量を減らしたいときには，結果の最適性を犠牲にする方法をとらなければならない．その一つは候補絞り込みであり，ある部分的組合せ結論 $B \in \mathcal{A}^{I'}(I' < I)$ について，B をその一部分とする組合せ結論 A のすべてのスコアの値が他の結論のスコアと比較して無視できるくらい小さいと経験的に判断される場合は，それらのスコア $\hat{S}_A(X;\Lambda)$ を最後まで計算することなく計算から除外してしまうという方法である．例えば，ある固定の $I' < I$ について，すべての $B \in \mathcal{A}^{I'}$ に候補絞り込み処理を施し，その数を $1/N$ に減らした場合は，総数を $1/N^{I-I'}$ に減らすことができるので，劇的な効果がある．

ある組合せ結論 A に至る部分的組合せ結論のうちで絞り込みの判定に用られるものの全体集合を $\Phi(A)$ とし，部分的組合せ結論 B による絞り込みの判定を関数 $\omega_B(X;\Lambda)$ で表すと，候補絞り込み関数 $\omega_B(X;\Lambda)$ は，B をその一部とする組合せ結論のスコアの値が無視できるくらい小さいと判定する場合は 0，そうでないとき 1 をとる関数である．このとき，候補絞り込み処理を含めた場合の組合せ結論の決定および分類は，スコア関数を

$$\tilde{S}_A(X;\Lambda) = \hat{S}_A(X;\Lambda) + \sum_{B \in \Phi(A)} \ln \omega_B(X;\Lambda) \tag{22}$$

および

$$\tilde{S}_c(X\,;\Lambda) = \frac{1}{\xi_C} \ln \sum_{A\in\Omega_c} \exp(\xi_C \tilde{S}_A(X\,;\Lambda)) \tag{23}$$

として,

$$\tilde{A}^* = \arg\max_{A\in\Omega} \tilde{S}_A(X\,;\Lambda) \tag{24}$$

$$\tilde{c}^* = \arg\max_{c} \tilde{S}_c(X\,;\Lambda) \tag{25}$$

なる $\tilde{A}^*$ および $\tilde{c}^*$ を選ぶことである. スコア関数 $\tilde{S}_A(X\,;\Lambda)$ の計算において, 棄却される組合せ結論の候補については, $\tilde{S}_A(X\,;\Lambda)$ の値が負の無限大となり最適解にはなり得ない, あるいは, $\exp(\tilde{S}_A(X\,;\Lambda))$ の値が 0 となり $\tilde{S}_c(X\,;\Lambda)$ の値に影響しない, と見なし, $\hat{S}_A(X\,;\Lambda)$ の値を求めない.

$\tilde{A}^*$ と A^*, $\tilde{c}^*$ と c^* は一般に一致しないが, 候補絞り込み関数 $\omega_B(X\,;\Lambda)$ の選択がその不一致の頻度と計算量に大きく影響を及ぼす. その点を考慮し, 通常, 候補絞り込み関数は経験的な知識に基づいて適当に選ばれる.

3 組合せ決定・分類のための最適化設計法

本節では, 式(22), (23)によるスコア $\tilde{S}_A(X\,;\Lambda)$, $\tilde{S}_c(X\,;\Lambda)$ に基づいて結論の組を決定あるいは分類する機構のための, パラメータ集合 Λ の設計法について検討する.

3.1 最大尤度規準に基づく設計

パラメータ集合 Λ からなる決定機構による結論 α_j の事後確率 $\Pr(\alpha_j \,|\, x)$ の推定値 $\pi_j(x\,;\Lambda)$ は, 式(6)により

$$\pi_j(x\,;\Lambda) = \frac{\exp(\eta_j(x\,;\Lambda))}{1+\exp(\eta_j(x\,;\Lambda))} \tag{26}$$

である. $p_j(x)$ を, 証拠 x にする結論 α_j が正しかったとき1, そうでなかったとき0を取る二値関数とすると, 証拠 x に対するパラメータ集合 Λ の対数尤度は,

$$\ell(x\,;\Lambda) = \ln \prod_{j=1}^{J} \pi_j(x\,;\Lambda)^{p_j(x)} \{1-\pi_j(x\,;\Lambda)\}^{1-p_j(x)} \tag{27}$$

$$= \sum_{j=1}^{J} \Big[p_j(x)\eta_j(x\,;\Lambda) - \ln\{1+\exp(\eta_j(x\,;\Lambda))\} \Big] \tag{28}$$

である. このとき, 尤度関数

$$L(\Lambda) = E[\,\ell(x\,;\Lambda)] \tag{29}$$

を最大化するパラメータ集合

$$\Lambda^* = \arg\max_{\Lambda} L(\Lambda) \tag{30}$$

は最大尤度規準のもとに最適である.

$\ell(X\,;\Lambda)$ が Λ に関して1次微分可能な連続関数であれば, 最急降下法や一般化確率的降下法を用いて Λ^* の準最適解を数値的に求めることができる.

この設計法は候補絞り込み処理に関しての考慮ができない. そのうえ, 一般に事後確率分布の形は不明であるので, このように事後確率分布形状を仮定したうえでの最尤推定に基づくパラメータ集合は与えられたデータに対する最小決定誤りを保証しない. 類ごとのデータの数にばらつきがある場合の非頑健性も指摘されている[10]. また, 組合せ分類においては各証拠 x_i ごとに $p_j(x_i)$ の値を確定することができないので, 最尤推定は難しい(EM アルゴリズムを使わなければならない). ただし, 3.2節あるいは3.3節における確率的勾配探索による設計法のための妥当な初期値を与える.

3.2 最小決定誤り規準に基づく設計

パラメータ集合設計における最も重要な目的は結果として得られる決定機構の決定誤りを最小化することであるので，本節では，決定誤りを直接最小化することを規準とする設計法を取り扱う．

正しい結論の組が A° であるとき，費用関数を

$$\ell(X\,;\Lambda) = 1\left(\tilde{S}_{A^\circ}(X\,;\Lambda) \leq \max_{A\neq A^\circ} \tilde{S}_A(X\,;\Lambda)\right) \tag{31}$$

とする．これは，式(13)による組合せ決定が誤っていたとき1，正しかったとき0をとる関数であるので，その期待値

$$L(\Lambda) = E[\,\ell(X\,;\Lambda)] \tag{32}$$

は組合せ決定誤り率を表す．$L(\Lambda)$ を損失関数としてそれを最小化するパラメータ集合

$$\Lambda^{**} = \arg\min_{\Lambda} L(\Lambda) \tag{33}$$

が組合せ決定誤りを最小化するパラメータ集合である．

分類問題の場合は，正しい分類結果を c° とすると，

$$\ell(X\,;\Lambda) = 1\left(\tilde{S}_{c^\circ}(X\,;\Lambda) \leq \max_{c\neq c^\circ} \tilde{S}_c(X\,;\Lambda)\right) \tag{34}$$

なる費用関数を用いる．

上の定義に基づく Λ^{**} を有限の標本から効果的に求める方法は知られていないので，費用関数を連続関数

$$\tilde{\ell}(X\,;\Lambda) = \tilde{1}\left(\tilde{S}_{A^\circ}(X\,;\Lambda) - \frac{1}{\zeta_A}\ln\frac{1}{|\Omega|-1}\sum_{A\neq A^\circ}\exp(\zeta_A\tilde{S}_A(X\,;\Lambda))\right) \tag{35}$$

で近似する．ただし，ζ_A は正の定数，演算子 $|\cdot|$ は集合の要素の総数を表し，関数 $\tilde{1}(\cdot)$ は，二値ステップ関数

$$1(y) = \begin{cases} 0, & y < 0 \\ 1, & 0 \geq y \end{cases} \tag{36}$$

を近似する1次微分可能な連続関数である(例えば，シグモイド関数 $\tilde{1}(x) = 1/\{1+\exp(-x)\}$)．以後，関数 $\tilde{1}(\cdot)$ を平滑化ステップ関数と呼ぶことにする．分類問題の場合は

$$\tilde{\ell}(X\,;\Lambda) = \tilde{1}\left(\tilde{S}_{c^\circ}(X\,;\Lambda) - \frac{1}{\zeta_C}\ln\frac{1}{C-1}\sum_{c\neq c^\circ}\exp(\zeta_C\tilde{S}_c(X\,;\Lambda))\right) \tag{37}$$

で近似する．ただし，ζ_C は正の定数である．$\tilde{\ell}(X\,;\Lambda)$ が Λ に関して1次微分可能な連続関数であれば，一般化確率的降下法を用いることができる．$\tilde{1}(y) \to 1(y)$ とすることで，近似費用関数をいくらでも真の費用関数に近付けることができる．

3.3 候補絞り込みの最適化設計

候補絞り込みの目的は，できるだけ少ない計算量でできるだけ正しい結論を得ることなので，候補絞り込み処理機構を含めた系全体は，決定誤り率の最小化だけではなく計算量の観点からも最適化されるべきである．

そこで，決定誤りあるいは分類誤りの費用関数 $\ell(X\,;\Lambda)$ を3.2節と同様に式(31)または(34)によって定め，その他に，

$$\ell'(X\,;\Lambda) = \sum_{B\in\Psi(X)} \omega_B(X\,;\Lambda) \tag{38}$$

なる費用関数を定義する．ただし，$\Psi(X)$ は証拠の組 X が得られたときに結論の組を得るときの候補絞り込みに使用する部分的結論 B の集合である．$\ell'(X;\Lambda)$ の期待値は絞り込みの甘さの度合を表しており，絞り込みが甘いほど計算量が多くなることが予想されるので，これを計算量の費用関数と呼ぶことにする．損失関数を，二つの費用関数の正の定数 γ による加重和の期待値

$$L'(\Lambda) = E[\ell(X;\Lambda) + \gamma\ell'(X;\Lambda)] \tag{39}$$

で定義するとき，この $L'(\Lambda)$ を最小化するパラメータ集合

$$\Lambda'^{**} = \arg\min_{\Lambda} L'(\Lambda) \tag{40}$$

は，決定誤りの数と計算量の両方の観点からの最適なパラメータ集合と見なすことができる．両者の均衡の度合は定数 γ によって制御できる．

二つの費用関数 $\ell(X;\Lambda)$ および $\ell'(X;\Lambda)$ の両方を，式 (35) と同様に平滑化ステップ関数を用いて Λ について一次微分可能な連続関数で近似すると，一般化確率的降下法を用いて Λ'^{**} の準最適数値解を求めることができる．

4 スポッティング型連続音声認識器への適用

本節では，組合せ決定機構の具体的事例として，スポッティングに基づく連続音声認識の枠組を取り上げる．

観測音声を $X = \{\boldsymbol{x}_i\}_{i=1}^{I}$ とする．ただし，各要素は S 次元実ベクトルである ($\boldsymbol{x}_i \in \Re^S$)．単語語彙が $\mathcal{W} = \{w_k\}_{k=1}^{K}$ であるとき，文法によって許される単語列の集合を G として，観測時系列 X を得たときにすべての単語列 $W_c = \{w_{k_c(1)}, w_{k_c(2)}, \cdots, w_{k_c(l_c)}\} \in G$($l_c$ は単語列 W_c の長さ) のうちで最も確からしい単語列 W_{c^*} を求めたい．

観測部分時系列 $X_s^e = \{\boldsymbol{x}_s, \boldsymbol{x}_{s+1}, \ldots, \boldsymbol{x}_{e-1}, \boldsymbol{x}_e\}(1 \le s \le e \le I)$ がどの単語に一致するかという結論 a_s^e のすべての s, e についての組合せ A のうちで，結果として単語列 W_c が得られるものの集合を Ω_c とすると，この問題は 2.3節で述べた組合せ分類問題であるので，必要に応じて 3節で述べた 3 種類のいずれかの最適化規準を用いてパラメータ集合 Λ'^{**} を設計することができる．本節では，その具体的な実装例を紹介する．

4.1 対数事後オッズ推定関数

観測部分時系列 X_s^e が単語 w_k である対数事後オッズが，パラメータ集合 Λ によって $Y_k(X_s^e;\Lambda)$ であると推定されるものとする．各単語 w_k はサブワード (例えば音素や音響イベント) 集合 $\mathcal{A} = \{\alpha_j\}_{j=1}^{J}$ の要素の連結として表現されるものとし，Λ はサブワードごとのモデル λ_j からなる ($\Lambda = \{\lambda_j\}_{j=1}^{J}$) とする．

各サブワードモデルは，プロトタイプ R_j と分散共分散行列集合 V_j と係数ベクトル $\boldsymbol{\phi}_j$ からなる ($\lambda_j = \{R_j, V_j, \boldsymbol{\phi}_j\}$) とする．ただし，プロトタイプは，$M$ 個の参照ベクトルの集合が N 個縦続に連結されたものとし，$R_j = \{R_{jn} = \{\boldsymbol{r}_{jnm}\}_{m=1}^{M}\}_{n=1}^{N}$，$V_j = \{V_{jn} = \{\Sigma_{jnm}\}_{m=1}^{M}\}_{n=1}^{N}$，$\boldsymbol{\phi}_j = \{\varphi_{j0}, \varphi_{j1}\}$ である．ここに，$\boldsymbol{r}_{jnm} \in \Re^S$，$\Sigma_{jnm} \in \Re^{S\times S}$．

まず，観測部分時系列 X_s^e がサブワード α_j である対数事後オッズがプロトタイプ R_j の分散共分散行列集合 V_j による距離 $D(X_s^e, R_j, V_j)$ の 1 次式によって推定されるモデルを想定し，

$$\eta_j(X_s^e;\Lambda) = \varphi_{j0} + \varphi_{j1} D(X_s^e, R_j, V_j) \tag{41}$$

と定義する．ここで，距離 $D(X_s^e, R_j, V_j)$ は次のように階層的に定義される．

第一に，観測音声の時刻 i のベクトル $\boldsymbol{x}_i$ と一参照ベクトル $\boldsymbol{r}_{jnm}$ との距離を，対応する分散共分散行列 Σ_{jnm} による二次形式によって

$$\delta(\boldsymbol{x}_i, \boldsymbol{r}_{jnm}, \Sigma_{jnm}) = (\boldsymbol{x}_i - \boldsymbol{r}_{jnm})^T \Sigma_{jnm}^{-1} (\boldsymbol{x}_i - \boldsymbol{r}_{jnm}) \tag{42}$$

と定義する．これを局所距離と呼ぶことにする．

第二に，観測音声の時刻 i のベクトル $\boldsymbol{x}_i$ とプロトタイプ R_j の n 番目の参照ベクトル集合 R_{jn} との距離を，

$$\Delta(\boldsymbol{x}_i, R_{jn}, V_{jn}) = \frac{1}{\xi_S} \ln \frac{1}{M} \sum_{m=1}^{M} \exp\left(\xi_S \delta(\boldsymbol{x}_i, \boldsymbol{r}_{jnm}, \Sigma_{jnm})\right) \tag{43}$$

と定義し，状態距離と呼ぶことにする．ただし，ξ_S は正の定数である．

第三に，観測部分時系列 X_s^e とプロトタイプ R_j の一つの対応経路における距離を考える．対応経路 $\theta = \{\iota_l, \nu_l\}_{l=1}^{L_\theta}$ は，$\{i, m\} = \{\iota_l, \nu_l\}$ によって時刻 $i = s, s+1, \ldots, e-1, e$ とプロトタイプの参照ベクトル集合の指標 $n = 1, 2, \ldots, N$ を対応づける二次元座標の集合で，端点条件 $\iota_1 = s$，$\iota_{L_\theta} = e$，$\nu_1 = 1$，$\nu_{L_\theta} = N$ と，順序条件 $\iota_l \leq \iota_{l+1}$，$\nu_l \leq \nu_{l+1}$ をすべて満たすものとする．このとき，各々の対応経路における距離を

$$D_\theta(X_s^e, R_j, V_j) = \sum_{l=1}^{L_\theta} \rho_l(\theta) \Delta(\boldsymbol{x}_{\iota_l}, R_{j\nu_l}, V_{j\nu_l}) \tag{44}$$

と定義する．ここに，$\rho_l(\theta)$ は対応経路 θ に依存する重み係数で $\sum_{l=1}^{L_\theta} \rho_l(\theta) = 1$ を満たし，Θ_s^e は区間$[s, e]$で考えられうる対応経路 θ の全体集合である．この距離 $D_\theta(\cdot)$ を経路距離と呼ぶことにする．

最後に，式 (41) の距離 $D(\cdot)$ を

$$D(X_s^e, R_j, V_j) = -\frac{1}{\xi_G} \ln \frac{1}{|\Theta_s^e|} \sum_{\theta \in \Theta_s^e} \exp\left(-\xi_G D_\theta(X_s^e, R_j, V_j)\right) \tag{45}$$

と定義し，以後，一般距離と呼ぶことにする．ただし，ξ_G は正の定数である．

次に，サブワードごとの対数事後オッズ $\eta_j(\cdot)$ を用いて，サブワード α_j の連結で表現された単語 $w_k = \{\alpha_{j_k(1)}, \alpha_{j_k(2)} \cdots, \alpha_{j_k(L_k)}\}$ の対数事後オッズ $Y_k(X_s^e; \Lambda)$ を次のように階層的に定義する．

第一に，観測部分時系列 X_s^e の一つサブワード境界列における対数事後オッズを考える．境界列 $\beta = \{b_l\}_{l=0}^{L_k}$ は，部分時系列 $X_{b_{l-1}}^{b_l}$ と単語 w_k の l 番目のサブワード $\alpha_{j_k(l)}$ を対応づけ，端点条件 $b_0 = s$，$b_{L_k} = e$ を満たすものとする．このとき，各々のサブワード境界列における単語 w_k の対数事後オッズを

$$y_{k\beta}(X_s^e; \Lambda) = \sum_{l=1}^{L_k} \eta_{j_k(l)}(X_{b_{l-1}}^{b_l}; \Lambda) \tag{46}$$

とする．

第二に，単語 w_k の対数事後オッズを，$y_{k\beta}(\cdot)$ を用いて

$$Y_k(X_s^e; \Lambda) = \frac{1}{\xi_W} \ln \sum_{\beta \in \mathcal{B}_s^e} \exp\left(\xi_W y_{k\beta}(X_s^e; \Lambda)\right) \tag{47}$$

と定義する．ただし，ξ_W は正の定数，$\mathcal{B}_s^e$ は X_s^e において考えられうる境界列 β の全体集合である．

4.2 単語列の認識

連続音声認識という問題の特性から，単語列を得るための決定の組合せの全体集合 Ω は，次の条件を満たす A の集合である．

1. 単語が存在するときは一つに決定しなければならないので，

$$\mathcal{P} = \{\emptyset, \{w_1\}, \{w_2\}, \ldots, \{w_K\}\}. \tag{48}$$

2. 時間的に隣接する単語の時間的重なりがあってはいけないので，

$$\forall\{a_{s_1}^{e_1} \in A, a_{s_2}^{e_2} \in A \,|\, s_1 \leq s_2 \leq e_1 \vee s_1 \leq e_2 \leq e_1\} : a_{s_1}^{e_1} = \emptyset \vee a_{s_2}^{e_2} = \emptyset. \quad (49)$$

このとき，条件1により $a_s^e \neq \emptyset$ となる部分的結論からは単語が一意に決まり，条件2により単語の時間的前後関係が明確になるので，単語列も一意に決定することができる．

前節のパラメータ集合により，部分的結論 a_s^e の組合せ A のスコアは

$$\hat{S}_A(X\,;\Lambda) = \sum_{s=1}^{I}\sum_{e=s}^{I}\sum_{k=1}^{K} 1(w_k \in a_s^e) Y_k(X_s^e\,;\Lambda) \quad (50)$$

で計算されるので，単語列 W_c のスコアはこの式と式(21)を用いて計算される．このように単語列のスコアとして対数事後オッズの累積を用いた場合は，単語列中に含まれる単語数に関係なく比較できることに注意して欲しい．

4.3 単語レベルの絞り込み

すべての部分観測時系列 X_s^e について $|\mathcal{P}| = J+1$ 通りの結論があるので，組合せ結論 A の総数は $O((J+1)^{I^2/2})$ と多い．そこで本節では，これを減らすために，単語レベルの絞り込みを導入する場合を考える．ここでは，一例として，次に述べる簡単で古典的な方法を採用する．すなわち，各単語について，

1. 近傍の部分観測時系列に対するスコアの中で最大値をとる
2. スコアの値がしきい値を超える

の両方を満たす部分観測時系列のみをその単語の候補として残す．部分観測時系列 X_s^e が単語 w_k に一致するという部分的結論を $B(w_k \,|\, X_s^e)$ と表記すると，上の条件に基づく絞り込みを含めた単語列のスコアは

$$\tilde{S}_A(X\,;\Lambda) = \hat{S}_A(X\,;\Lambda) + \sum_{1\leq s\leq e\leq I} \omega_{B(w_k\,|\,X_s^e)}(X\,;\Lambda) \quad (51)$$

と表される．$\omega_{B(w_k\,|\,X_s^e)}(X\,;\Lambda)$ は候補絞り込み関数であり，

$$\omega_{B(w_k\,|\,X_s^e)}(X\,;\Lambda) = \omega_{1k}(X_s^e\,;\Lambda)\cdot\omega_{2k}(X_s^e\,;\Lambda) \quad (52)$$

と定義される．ここに，$\omega_{1k}(\cdot)$，$\omega_{2k}(\cdot)$ はそれぞれ条件1，2に対応し，

$$\begin{aligned}\omega_{1k}(X_s^e\,;\Lambda) &= 1\left(Y_k(X_s^e\,;\Lambda) - \max_{s'\in\mathcal{S}_k(e)} Y_k(X_{s'}^e\,;\Lambda)\right)\cdot \\ &\quad 1\left(Y_k(X_s^e\,;\Lambda) - \max_{1\leq|e'-e|\leq\kappa_k}\max_{s'\in\mathcal{S}_k(e')} Y_k(X_{s'}^{e'}\,;\Lambda)\right) \quad (53)\\ \omega_{2k}(X_s^e\,;\Lambda) &= 1\left(Y_k(X_s^e\,;\Lambda) - h_k\right) \quad (54)\end{aligned}$$

とする．ただし，κ_k は定数，h_k はしきい値，$\mathcal{S}_k(e)$ は終端 e に対して単語 w_k の始端 s が取りうる値の集合であり，$\Lambda = \{\{\lambda_j\}_{j=1}^{J}, \{h_k\}_{k=1}^{K}\}$．$\omega_{1k}(\cdot)$ は，探索の効率の観点から，「近傍の部分観測時系列の中でスコアが最大」という条件を，始端に関する最大条件と終端に関する最大条件の2段階に分けて表現している．3.3節の最適化を適用するために，関数 $\omega_{B(w_k\,|\,X_s^e)}(X\,;\Lambda)$ を連続関数で近似すると，

$$\omega_{B(w_k\,|\,X_s^e)}(X\,;\Lambda) \approx \tilde{\omega}_{1k}(X_s^e\,;\Lambda)\cdot\tilde{\omega}_{2k}(X_s^e\,;\Lambda) \quad (55)$$

ここに，

$$\tilde{\omega}_{1k}(X_s^e;\Lambda) = \tilde{\mathrm{I}}\left(Y_k(X_s^e;\Lambda)-\chi_{ek}(X;\Lambda)\right)\cdot \tilde{\mathrm{I}}\left(Y_k(X_s^e;\Lambda)-\frac{1}{\xi_E}\ln\frac{1}{2\kappa_k-1}\sum_{1\le|e'-e|\le\kappa_k}\exp(\xi_E\chi_{e'k}(X;\Lambda))\right) \quad (56)$$

$$\tilde{\omega}_{2k}(X_s^e;\Lambda) = \tilde{\mathrm{I}}\left(Y_k(X_s^e;\Lambda)-h_k\right) \quad (57)$$

ただし，ξ_B，ξ_E はともに正の定数であり，

$$\chi_{ek}(X;\Lambda) = -\frac{1}{\xi_B}\ln\frac{1}{|\mathcal{S}_k(e)|}\sum_{s'\in\mathcal{S}_k(e)}\exp\left(-\xi_S Y_k(X_{s'}^e;\Lambda)\right) \quad (58)$$

である．

4.4 実装のための工夫

4.1節から4.3節にわたり，ワードスポッティングに基づく連続音声認識器の一実装例を比較的一般化された形で示した．この系に3節の最適化法を適用すれば，望む性格の認識器を設計することは理論的には可能なはずである．しかしながら，我々に現実に与えられているのは限られた計算機資源と限られた学習用標本のみであり，実際には様々な問題に直面する．この節では，それらの問題のための現実的な対処のしかたを検討する．

第一に，限られた計算量で認識および学習を実現することを考えよう．まず，対数事後オッズ推定関数によるスコアの定義において，我々は式(21)のような形を多用した．前述のとおり，この値を求めるには非常に多くの計算量を要する．そこで，必要に応じて ξ_C などの定数の正の無限大極限，すなわち最大値や最小値で代用する．こうすることで動的計画法などによる計算量の削減が可能である．また，式(52)による候補絞り込み関数の定義に現れる平滑化ステップ関数 $\tilde{\mathrm{I}}(\cdot)$ として例えば例に挙げたシグモイド関数を用いると，その値は常に正なので，本来は候補絞り込みにおいて棄却されるはずのすべての単語仮説を学習時には棄却せずに残しておかなくてはならない．そこで，しきい値から遠いところでは恒等的に値0を取る区分線形関数

$$\tilde{\mathrm{I}}(x;\alpha) = \begin{cases} 0 & \text{if } x \le -\dfrac{1}{2\alpha}, \\ \dfrac{1}{2}(\alpha x+1) & \text{if } -\dfrac{1}{2\alpha} < x \le \dfrac{1}{2\alpha}, \\ 1 & \text{if } \dfrac{1}{2\alpha} < x. \end{cases} \quad (59)$$

や区分放物線関数

$$\tilde{\mathrm{I}}(x;\alpha) = \begin{cases} 0 & \text{if } x \le -\dfrac{1}{\alpha}, \\ \dfrac{1}{2}(\alpha x+1)^2 & \text{if } -\dfrac{1}{\alpha} < x \le 0, \\ 1-\dfrac{1}{2}(\alpha x-1)^2 & \text{if } 0 < x \le \dfrac{1}{\alpha}, \\ 1 & \text{if } \dfrac{1}{\alpha} < x. \end{cases} \quad (60)$$

で代用する．

第二に，限られた数の学習標本から最適なパラメータを推定しなければならない現実を鑑みると，自由パラメータの数はできるだけ少なくするように工夫しなければならない．例えば，サブワードモデルにおける分散共分散行列 Σ_{jmn} は自由パラメータ数が多いので，単位行列に固定したり，対角行列にして対角成分のみを自由パラメータとするなどである．特に，独立性が低いと思われるパラメータを「結び」の関係にすることが有効であろう．予備選択のしきい

値 h_k を単語によらず共通の値にすることや，すでに述べたように全ての単語のモデルを少ない種類のサブワードの連結で表現することなどである．

第三に，本稿で提案した方法は勾配探索によっているので，収束が極端に遅くならないようにそれぞれのパラメータの歩み幅の均衡に注意する必要がある．また，パラメータの変域にも注意しなければならない．例えば，φ_{j1} の変域は $(-\infty, 0)$ であるので，勾配探索の定義による微修正によって許された変域からはみ出してしまうことがある．このような場合，例えば $\varphi_{j1} = -\exp(\varphi'_{j1})$ と置き換え，φ'_{j1} を変域 $(-\infty, \infty)$ の自由パラメータとすればよい．あるいは，最初から歩み幅の調整の難しいパラメータは使わないように系を構成してもよい．例えば，4.1節で用いた式 (41) の代わりに，係数ベクトル ϕ_j をパラメータとして用いないで，近隣の類あるいは前述の「ごみ類」との競合に基づく確からしさとして

$$\eta_j(X_s^e; \Lambda) = -D(X_s^e, R_j, V_j) - \frac{1}{\zeta_D} \ln \frac{1}{J-1} \sum_{j' \neq j} \exp(-\zeta_D D(X_s^e, R_{j'}, V_{j'})) \qquad (61)$$

または

$$\eta_j(X_s^e; \Lambda) = -D(X_s^e, R_j, V_j) + D(X_s^e, R_0, V_0) \qquad (62)$$

などと定義できる．ただし，ζ_D は正の定数，$\lambda_0 = \{R_0, V_0\}$ は「ごみ類」のモデルである．McDermott らの実装 [8] は式 (62) に基づく単語列スコアの最大値を時間同期的に探索するものと等価である．

本節で述べた現実的実装のための工夫は経験則に頼るところが大きく，今後のさらなる研究が望まれる．

5 まとめ

不確実性を伴う組合せ決定問題を解く枠組として，仮説の対数事後オッズの累積に基づくものを定式化し，計算量や記憶容量の節約のための絞り込み処理も含めた枠組での決定誤りの数と計算量を最小化する設計法を提案した．この新しい方法の応用例として，単語スポッティングに基づく連続音声認識の枠組の場合を紹介し，実装のための工夫も検討した．応用は音声認識のみに留まらず，不確実性を伴う推論問題一般に適用できると考えられる．また，音声認識に限っても，対数事後オッズ推定関数や候補絞り込みの規準には数多くの選択肢がある．それらの決定は様々な条件を考慮して注意深くなされるべきである．

本稿で取り扱った決定機構は，一つの証拠から導かれ得る複数の結論についてそれぞれ独立評価したスコアを用いるものであり，並列分散型の計算機への実装に適している．

参考文献

[1] 速水，岡，“連続 DP による連続単語認識実験とその考察，” 信学論誌，Vol. J67-D, No.6, pp.677–684 (1984).

[2] 中川，“拡張連続 DP 法による連続音声認識，” 信学論誌，Vol. J67-D, No.6, pp.677–684 (1984).

[3] 川端，花沢，鹿野，“HMM 音韻認識に基づくワードスポッティング，” 信学技報，SP88-23, pp.17–22 (1988).

[4] H. Tsuboi and Y. Takebayashi, “A Real-Time Task-Oriented Speech Understanding System Using Keyword-Spotting,” IEEE, Proc. of ICASSP-92, Vol.1, pp.197–200 (1992).

[5] J. G. Wilpon, L. R. Rabiner, C.-H. Lee, and E. R. Goldman, “Automatic Recognition of Keywords in Unconstrained Speech Using Hidden Markov Models,” IEEE Trans. on ASSP, Vol. 38, No. 11, pp.1870–1878 (1990).

[6] S. Katagiri, C.-H. Lee, and B.-H. Juang, "A Generalized Probabilistic Descent Method," Proc. of ASJ Fall Meeting, 2-P-6, pp.141–142 (1990).

[7] S. Katagiri, C.-H. Lee, and B.-H. Juang, "New Discriminative Training Algorithms Based on the Generalized Probabilistic Descent Method," Proc. of the 1991 IEEE Workshop on Neural Networks for Signal Processing, pp.299–308 (1991).

[8] E. McDermott and S. Katagiri, "Prototype-Based MCE/GPD Training for Word Spotting and Connected Word Recognition," IEEE, Proc. of ICASSP-93, Vol.2, pp.291–294 (1993).

[9] T. Komori and S. Katagiri, "A New Learning Algorithm for Minimizing Spotting Errrors," Proc. of the 1993 IEEE Workshop on Neural Networks for Signal Processing, pp.333–342 (1993).

[10] H. Gish, "A Minimum Classification Error, Maximum Likelihood, Neural Networks," IEEE, Proc. of ICASSP-92, Vol.2, pp.289–292 (1992).

A Novel Spotting-Based Approach to Continuous Speech Recognition: Minimum Error Classification of Keyword-Sequences

Takashi Komori and Shigeru Katagiri

Abstract

To overcome the lack of theoretical basis of a fundamental, word spotting-based approach to the recognition of natural, spontaneous speech utterances, we propose in this paper a novel *spotter* (spotting system) design method referred to as Minimum Error Classification of Keyword-sequences (MECK). A key concept of the method is to formalize the entire spotting process as a trainable functional form with the design objective being the *keyword-sequence* (a string of prescribed keyword categories) classification accuracy. A resulting MECK procedure allows one to design spotters in an efficient way of using only pairs of utterances and their corresponding phonemic transcriptions (not requiring hand-segmented labels) as well as in a mathematically-proven way consistent with the error minimization of the keyword-sequence classification. MECK is quite general and can be applied to any reasonable spotter structure. The paper specially presents implementation details for a prototype-based spotter and demonstrates the utility of this MECK-trained spotter in several Japanese keyword spotting tasks.

1 Introduction

The recognition of natural and spontaneous (unconstrained in terms of speaking style and condition) speech utterances is an important issue for realizing a user-friendly human-machine interface. Since natural speech often contains various ill-conditioned phenomena, such as hesitations, repetitions, filled pauses, noises including lip noise and background noise, and interjections, that are usually difficult to describe (model) in a word-by-word mode with a written language grammar, it has been considered that a conventional, word-by-word recognition approach, which was widely used for the recognition of continuous

read sentences, is insufficient to handle such natural utterances appropriately. Recently, an alternative to this conventional approach, namely keyword spotting, has been studied with increasing vigor [1]–[10]. This paper is intended to propose a novel method of designing a system of executing this spotting process, namely a *spotter*.

In nature, the goal of spotting is to correctly spot (detect) all of the included keywords from an input utterance; in other words, to correctly classify the input as one of the possible keyword-sequences, each consisting of only prescribed keyword class names. The keyword-sequence itself can be an input to a post-end process such as context modeling or semantic modeling. Spotter performance should thus be evaluated by the classification accuracy of the keyword-sequences instead of the accuracy of individual spotting decisions. In other words, spotters should be designed in a consistent manner with the minimum error status of keyword-sequence classification. However, as seen in literature [1]–[10], recent efforts have actually been made in light of the error reduction of individual spotting decisions and also been merely heuristic, entailing no optimality in the sense of minimum error classification of keyword-sequences.

In light of this, we propose in this paper a novel design method for spotters. We refer to the method in the paper as Minimum Error Classification of Keyword-sequences (MECK). A key concept of this method is to formalize the spotting process as a trainable functional form with the design objective being the keyword-sequence classification accuracy. In particular, MECK embeds the spotting process in a *smooth* (at least the first differentiable in spotter adjustable parameters) functional form; it formulates an individual keyword spotting operation as a two-class segmentation/classification by using the discriminant function based on *a posteriori* odds for this classification; it also introduces a mathematically proven, GPD-based optimization algorithm to achieve the *optimal* (minimum keyword-sequence classification error) status of the spotter.

MECK is quite general and can be applied to any reasonable spotter structure. By way of example, we present design details for a prototype-based spotter of which measurement calculation is simple and suited for a high-speed parallel computation implementation, and evaluate this prototype-based spotter in several Japanese keyword spotting tasks.

2 Definition

2.1 Problem formalization

Classification is a simple process to assign one of the possible classes to a given pattern, and it does not include a process to segment (extract) the pattern from its background's wider or larger signal. Similarly,

spoken word classification is defined as a process to classify a word segment pre-segmented from a continuous speech utterance as one of the possible word classes. Obviously, this simple-minded classification is insufficient for continuous speech recognition. The recognition process of continuous speech requires an appropriate link of segmentation and classification. Spotting can be considered the very framework to achieve this link directly. However, in reality, even in this promising framework, these two processes are designed separately, entailing no guarantee of the resulting spotting optimality. A main effort in our formalization is therefore to embed this complicated process in a unified functional form that is suited for the use of mathematically proven optimization techniques.

For clarity of presentation, in addition to the term of keyword-sequence classification (*word-sequence classification* for short), we define here the following two terms: 1) *keyword-sequence spotting* (*word-sequence spotting* for short) being used to decide whether a sequence of keywords is included in a given utterance and the location of these keywords, and 2) *keyword-spotting* (*word-spotting* for short) being used to decide whether a keyword exists in a preset segment.

Assume that our task is to classify a given speech utterance as one of C possible keyword-sequence classes, each consisting of only prescribed keyword names. Each utterance X is represented in the form of an acoustic feature vector sequence; $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_i, \ldots, \boldsymbol{x}_I\}$, where $\boldsymbol{x}_i$ is an S-dimensional acoustic feature vector. We denote the entire set of possible keyword-sequence classes by Ω; $\Omega = \{\Omega_1, \ldots, \Omega_c, \ldots, \Omega_C\}$, where Ω_c is the c-th keyword-sequence class. We also denote the entire set of prescribed keywords by W; $W = \{w_1, \ldots, w_k, \ldots, w_K\}$, where w_k is the k-th keyword.

Let us focus on word w_k's spotting decision in the segment X_s^e; $X_s^e = \{\boldsymbol{x}_s, \boldsymbol{x}_{s+1}, \ldots, \boldsymbol{x}_{e-1}, \boldsymbol{x}_e\}$. We denote this decision for w_k as a_{kse}. The a_{kse} is a kind of indicator function that becomes one (1) for a correct decision and zero (0) for an incorrect decision. This indicator function can be defined in principle for all s, e, and k. Nevertheless, in reality, the functions are defined in only a limited number of cases, due to several realistic restrictions on the indexes, such as k. We denote the entire set of possible combinations of s, e, and k by $\mathcal{G}$.

In this view, one word-sequence classification decision can be considered a sequence of word spotting decisions $\alpha = \{a_{kse}\}$, each corresponding to a spotted word class included in a word-sequence class, e.g., Ω_c. Therefore, the goal of the spotter design should be to achieve a state of adjustable spotter parameters, denoted by Λ, that emulates the following Bayesian decision theory-based rule [11]:

$$c(X) = \hat{c} \quad \text{if } \hat{c} = \arg\max_c \Pr(\Omega_c \,|\, X), \tag{1}$$

where $\Pr(\Omega_c \,|\, X)$ is the *a posteriori* probability of the word-sequence class Ω_c, given X, and $c(X)$ denotes the operation of making a word-

sequence classification decision. Note that this $c(X)$ is known to lead to the minimum error classification.

However, the above rule is too sophisticated and abstract. To define our design algorithm in a practical and effective fashion, we embody several operations/concepts in a mathematical form.

First, we approximate the *a posteriori* probability of a word-sequence class $\Pr(\Omega_c \mid X)$ by the *a posteriori* probability of the dominant (most probable) word-sequence spotting in the class:

$$\Pr(\Omega_c \mid X) \approx \max_{\alpha \in \mathcal{A}_c} \Pr(\alpha \mid X), \tag{2}$$

where $\mathcal{A}_c (\subset \mathcal{G})$ is a set of word-sequence spotting decisions, each corresponding to Ω_c, and $\Pr(\alpha \mid X)$ is the *a posteriori* probability of α, given X. Accordingly (1) becomes equivalent to

$$c(X) = \hat{c} \quad \text{if} \quad \arg\max_{\alpha \in \mathcal{A}_{\hat{c}}} \Pr(\alpha \mid X) = \hat{\alpha}, \tag{3}$$

where

$$\hat{\alpha} = \arg\max_{\alpha \in \mathcal{G}} \Pr(\alpha \mid X). \tag{4}$$

This rule more closely represents an actual spotting procedure.

Second, on the assumption that all of the individual *a posteriori* probabilities $\Pr(w_k \mid X_s^e)$ are independent of each other, we represent $\Pr(\alpha \mid X)$ as

$$\Pr(\alpha \mid X) = \prod_{k,s,e} \Pr(w_k \mid X_s^e)^{a_{kse}} \{1 - \Pr(w_k \mid X_s^e)\}^{1 - a_{kse}}, \tag{5}$$

and

$$\ln \Pr(\alpha \mid X) = \sum_{k,s,e} a_{kse} \ln \frac{\Pr(w_k \mid X_s^e)}{1 - \Pr(w_k \mid X_s^e)} + \sum_{k,s,e} \ln\{1 - \Pr(w_k \mid X_s^e)\}. \tag{6}$$

Now, $\Pr(\alpha \mid X)$ is represented by a set of more elemental *a posteriori* probabilities. However, in practice, even these *a posteriori* probabilities are rarely known and thus we must further replace these probabilities with some proper estimate that is a function of Λ. Note here that the first term on the right-side of (6) includes the *a posteriori* odds

$$O(w_k \mid X_s^e) = \frac{\Pr(w_k \mid X_s^e)}{1 - \Pr(w_k \mid X_s^e)} \tag{7}$$

that have been widely used in artificial intelligence and statistics. Naturally, we then use a *keyword possibility score*, denoted by $\eta_\Lambda(w_k \mid X_s^e)$, as the estimate of the logarithmic *a posteriori* odds. Note that this estimate is a function of Λ. Moreover, for the purpose of finding $\hat{\alpha}$, we can ignore the second term of the right side of (6) that always takes a constant value. Therefore, we can simplify (4) to

$$\hat{\alpha} = \arg\max_{\alpha \in \mathcal{G}} Y_\Lambda(\alpha \mid X), \tag{8}$$

where

$$Y_\Lambda(\alpha \mid X) = \sum_{k,s,e} a_{kse}\eta_\Lambda(w_k \mid X_s^e), \tag{9}$$

and accordingly rewrite (1) to

$$c(X) = \hat{c} \quad \text{if } \hat{c} = \arg\max_c g_\Lambda^c(X), \tag{10}$$

where $g_\Lambda^c(X)$ is defined as

$$g_\Lambda^c(X) = \frac{1}{\xi}\ln\left\{\frac{1}{|\mathcal{A}_c|}\sum_{\alpha\in\mathcal{A}_c}\exp(\xi Y_\Lambda(\alpha \mid X))\right\}, \tag{11}$$

ξ is a positive constant, and $Y_\Lambda(\alpha \mid X)$ is referred to as a *word-sequence spotting score* for α. Note that $g_\Lambda^c(X)$, which is referred to as the discriminant function for Ω_c, expresses an aggregate possibility that X includes Ω_c in a general form.

Consequently, in our design approach, (11) is used in place of $\Pr(\Omega_c \mid X)$; similarly, (10) is used in place of (1). Now it turns out that finding the optimal, concerning (11), status of Λ is our design target.

As in conventional classifier designs, there are two main approaches to the emulation of (10) (the design of Λ): 1) the maximum-likelihood design, and 2) the discriminant function method. Taking account of the findings of MCE/GPD studies [12],[13], we chose to use the second approach. Therefore, we next define the loss function

$$\ell(X;\Lambda) = 1\left(-g_\Lambda^{c^*}(X) + \max_{c\neq c^*} g_\Lambda^c(X)\right), \tag{12}$$

where

$$1(x) = \begin{cases} 0, & \text{if } x < 0 \\ 0.5, & \text{if } x = 0 \\ 1, & \text{if } x > 0, \end{cases} \tag{13}$$

and c^* is the correct word-sequence class index. This loss is zero (0) for a correct word-sequence classification; otherwise one (1). This loss represents an ideal error count. However, note here that the loss is discontinuous in Λ and causes mathematical problems in formalization as discussed in [12]. In our approach based on the MCE/GPD concept, we therefore use a smooth loss defined as

$$\ell(X;\Lambda) = \tilde{1}\left(-g_\Lambda^{c^*}(X) + \frac{1}{\zeta}\ln\left\{\frac{1}{C-1}\sum_{c\neq c^*}\exp(\zeta g_\Lambda^c(X))\right\}\right), \tag{14}$$

where $\tilde{1}(\cdot)$ is a smooth step function such as $\tilde{1}(x) = (1+\exp(-x/\varsigma))^{-1}$ with ς being a positive constant and ζ a positive constant.

In principle, the loss must be evaluated over all of the possible samples. We thus introduce the expected loss

$$L(\Lambda) = E_X[\,\ell(X;\Lambda)\,] \tag{15}$$

as the design objective to be minimized, where E_X is the expectation over the X-space.

2.2 Focusing in design: computation reduction

The design problem is now formalized as the minimization problem of the expected loss. Removing the second term of the right side from (6) actually contributes to reducing the computation. However, full computation of $Y_\Lambda(\alpha \mid X)$ is hopelessly time-consuming. There is still a clear need for further reduction of the computation. A natural way of such attempt is to focus the design efforts on plausible word-spotting decisions (remove probably incorrect decisions beforehand) in computating $Y_\Lambda(\alpha \mid X)$. In light of this, we introduce a pruning function $\omega_\Lambda(w_k \mid X_s^e)$ that indicates zero (0) when the corresponding word- spotting decision should be ignored and one (1) otherwise. Then $Y_\Lambda(\alpha \mid X)$ is replaced with the following practical version of the score:

$$\tilde{Y}_\Lambda(\alpha \mid X) = \sum_{k,s,e} a_{kse} \{\eta_\Lambda(w_k \mid X_s^e) + \ln \omega_\Lambda(w_k \mid X_s^e)\}. \qquad (16)$$

Note that $\omega_\Lambda(w_k \mid X_s^e)$ being zero makes the above practical score $\tilde{Y}_\Lambda(\alpha \mid X)$ negative infinity, which means that the corresponding word-spotting decision does not contribute to computing the sequence-spotting score, and that $\tilde{Y}_\Lambda(\alpha \mid X)$ can be substituted for $Y_\Lambda(\alpha \mid X)$ in (10) and (11) without any serious mathematical drawback.

The above way should markedly reduce the computation in the design stage. However, in many realistic large-vocabulary cases, even a reduced computation of the loss is still resource-consuming because the large number of possible word-sequences increases the number of word-spotting decisions explosively. Conventionally, there has been an attempt to solve this computation explosion problem in a separate, heuristic manner, such as the beam-search algorithm. Obviously, this heuristic approach does not allow one to achieve a consistent minimization of the loss. In light of this, we also consider another attempt of reduction by introducing the loss

$$\tilde{\ell}(X\,;\Lambda) = \sum_{k,s,e} \omega_\Lambda(w_k \mid X_s^e) \qquad (17)$$

that approximates the number of keyword hypotheses, each remaining (being not pruned) in the design process. If this number of keyword hypotheses is counted as loss (cost) in the same manner as (12), one can reduce the training computation by explicitly attempting to decrease this count; in other words, explicitly attempting to prune the keywords in the loss minimization process. Consequently, incorporating this new loss, the expected loss can also be re-defined as

$$\tilde{L}(\Lambda) = E_X\,[\,\ell(X\,;\Lambda) + \gamma\tilde{\ell}(X\,;\Lambda)\,], \qquad (18)$$

where γ is a controllable weighting factor.

2.3 Optimization algorithm

In accordance with the adaptive adjustment rule of GPD, we use

$$\Lambda_{t+1} = \Lambda_t - \epsilon_t U \nabla_\Lambda \ell(X_t; \Lambda_t), \tag{19}$$

which has been shown to lead to the optimal state of Λ that corresponds to at least the local minimum of the expected loss, where Λ_t denotes the parameter set at the t-th iteration, ϵ_t is a learning factor that satisfies $\sum_{t=1}^{\infty} \epsilon_t = \infty$ and $\sum_{t=1}^{\infty} \epsilon_t^2 < \infty$, U is a positive definite matrix, ∇_Λ is the gradient symbol with respect to Λ, and X_t denotes the t-th observation sequence given randomly for training.

The entire MECK formalization has been defined. It is worth noting that MECK allows one to design spotters by using only design speech utterances and their corresponding phonemic transcriptions. As seen above, MECK does not require hand-segmented labels for design.

3 Implementation

There are many ways to apply the proposed method to spotter design. MECK can be applied to any reasonable spotter structure such as a prototype-based system or an HMM system. Each keyword can be either directly modelled (represented in spotter parameters) or indirectly modelled by concatenating subword models. Among these many choices, this paper specially presents an implementation example for a prototype-based spotter consisting of subword models; λ_j ($\subset \Lambda$) denotes a class j subword model consisting of a sequence of acoustical feature vectors.

3.1 Log Estimate of A Posteriori Odds

Due to our preference for using subword models, we shall first define a subword-based log *a posteriori* odds estimate. Our prototype-based spotter basically computes the distance between the subword model and a speech segment. We thus need to convert this distance measure to the *a posteriori* odds form. Among many possible ways of doing so, we use the following function form:

$$\eta_\Lambda(\lambda_j \mid X_s^e) = \phi_{j0} + \phi_{j1} D(X_s^e, \lambda_j), \tag{20}$$

where $D(X_s^e, \lambda_j)$ is the distance between X_s^e and λ_j, and ϕ_{j0} and ϕ_{j1} are constants. The use of this form is motivated by the estimation of $\Pr(\lambda_j \mid X_s^e)$ using a logistic function of $D(X_s^e, \lambda_j)$. Here, we also assume that each subword model λ_j consists of a prototype R_j and

its corresponding covariance matrix set V_j; similar to HMM, R_j has an N-state structure and a set of reference vectors, each in the same vector space with the acoustical feature vector, that are assigned to each state, i.e.,

$$R_j = \{R_{jn}\}_{n=1}^{N}, \quad R_{jn} = \{\boldsymbol{r}_{jnm}\}_{m=1}^{M}, \quad \boldsymbol{r}_{jnm} \in \Re^S, \tag{21}$$

where R_{jn} is the reference vector set at the n-th state of the subword j model, and $\boldsymbol{r}_{jnm}$ is the S-dimensional m-th reference vector of R_{jn}; V_j consists of N state sets of covariance matrices, i.e.,

$$V_j = \{V_{jn}\}_{n=1}^{N}, \quad V_{jn} = \{\Sigma_{jnm}\}_{m=1}^{M}, \quad \Sigma_{jnm} \in \Re^{S\times S}, \tag{22}$$

where V_{jn} is the covariance matrix set at the n-th state of the subword j, and Σ_{jnm} is the $S \times S$ covariance matrix corresponding to $\boldsymbol{r}_{jnm}$. Then we define the distance $D(X_s^e, \lambda_j)$ in the following step-by-step manner.

First, we introduce a *local distance* between the i-th acoustic feature vector of X, $\boldsymbol{x}_i$, and a reference vector as

$$\delta(\boldsymbol{x}_i, \boldsymbol{r}_{jnm}, \Sigma_{jnm}) = (\boldsymbol{x}_i - \boldsymbol{r}_{jnm})^T \Sigma_{jnm}^{-1} (\boldsymbol{x}_i - \boldsymbol{r}_{jnm}). \tag{23}$$

Second, we introduce a *state distance* to $\boldsymbol{x}_i$ as

$$\Delta(\boldsymbol{x}_i, R_{jn}, V_{jn}) = -\frac{1}{\xi_S} \ln \frac{1}{M} \sum_{m=1}^{M} \exp\left(-\xi_S \delta(\boldsymbol{x}_i, \boldsymbol{r}_{jnm}, \Sigma_{jnm})\right), \tag{24}$$

where ξ_S is a positive constant. Third, we introduce a *path distance* using a DTW matching path between the observation subsequence X_s^e and λ_j. Consider a path $\theta = \{\iota_l, \nu_l\}_{l=1}^{L_\theta}$ that is a set of index pairs $\{\iota_l, \nu_l\}$, where $\iota_l \in [\,s, e\,]$, $\nu_l \in [\,1, N\,]$, $\iota_1 = s$, $\iota_{L_\theta} = e$, $\nu_1 = 1$, $\nu_{L_\theta} = N$, $\iota_l \leq \iota_{l+1}$, and $\nu_l \leq \nu_{l+1}$. Then the path distance is defined as

$$D_\theta(X_s^e, \lambda_j) = \sum_{l=1}^{L_\theta} \rho_{\theta l} \Delta(\boldsymbol{x}_{\iota_l}, R_{j\nu_l}, V_{j\nu_l}), \tag{25}$$

where $\rho_{\theta l}$ is a weighting factor that satisfies $\sum_{l=1}^{L_\theta} \rho_{\theta l} = 1$. Accordingly, the distance $D(X_s^e, \lambda_j)$ is defined as

$$D(X_s^e, \lambda_j) = -\frac{1}{\xi_G} \ln \frac{1}{|\Theta_s^e|} \sum_{\theta \in \Theta_s^e} \exp\left(-\xi_G D_\theta(X_s^e, \lambda_j)\right), \tag{26}$$

where Θ_s^e is the entire set of all possible matching paths between X_s^e and λ_j.

Next, we shall define a word-level log *a posteriori* odds estimate, $\eta_\Lambda(w_k \,|\, X_s^e)$, using the above subword-based estimates. This definition is again done in a step-by-step manner. Let a w_k keyword (word) model be represented by the sequence of L_k subword models, namely

$\{\lambda_{j_k,1}, \lambda_{j_k,2}, \ldots, \lambda_{j_k,L_k}\}$. Also let us consider a set of subword boundaries $\beta = \{b_l\}_{l=0}^{L_k}$, where b_l represents a matching boundary between the observation subsequence $X_{b_{l-1}+1}^{b_l}$ and the l-th subword $\lambda_{j_k,l}$ of keyword w_k; $b_0 = s-1$ and $b_{L_k} = e$. Thus, we first introduce a log *a posteriori* odds estimate for β as

$$\eta_\Lambda(w_k \mid X_s^e, \beta) = \sum_{l=1}^{L_k} \eta_\Lambda(\lambda_{j_k(l)} \mid X_{b_{l-1}+1}^{b_l}). \tag{27}$$

Then, we define $\eta_\Lambda(w_k \mid X_s^e)$ as

$$\eta_\Lambda(w_k \mid X_s^e) = \frac{1}{\xi_W} \ln \frac{1}{|B_s^e|} \sum_{\beta \in B_s^e} \exp\left(\xi_W \eta_\Lambda(w_k \mid X_s^e, \beta)\right), \tag{28}$$

where ξ_W is a positive constant and B_s^e is the entire set of all possible subword boundary sets.

3.2 Pruning Function

Needless to say, pruning always suffers from the risk of a decrease in accuracy because it often misses some of the correct word-spotting hypotheses. Therefore, the pruning function must be designed carefully. Our actual pruning strategy is summarized as follows; i.e., $\omega_\Lambda(w_k \mid X_s^e)$ is set closer to zero, if one of the following manifolds is correct.

1. $X_{s'}^e$ ($s' \neq s$) is more likely than X_s^e for spotting w_k.
2. $X_{s'}^{e'}$ (s' is arbitrary, $e' \neq e$, and $e' \approx e$) is more likely than X_s^e for spotting w_k.
3. $\eta_\Lambda(w_k \mid X_s^e)$ is less than a preset threshold.

This pruning criterion has actually been used in conventional spotting techniques using starting-end-free dynamic programming; which means that the criterion is not so specific to our method. However, based on our policy of formalizing the process rigorously, we too attempt to embed the criterion in the functional form design procedure here.

We first introduce three continuous auxiliary pruning functions, $\omega_\Lambda^1(w_k \mid X_s^e)$, $\omega_\Lambda^2(w_k \mid X_s^e)$, and $\omega_\Lambda^3(w_k \mid X_s^e)$, so as to approximate the above three conditions in a smooth functional form.

$$\omega_\Lambda^1(w_k \mid X_s^e) = \tilde{1}(\eta_\Lambda(w_k \mid X_s^e) - \chi_\Lambda(w_k \mid X, e)), \tag{29}$$

where

$$\chi_\Lambda(w_k \mid X, e) = \frac{1}{\zeta_S} \ln \frac{1}{|S_k(e)|} \sum_{s' \in S_k(e)} \exp\left(\zeta_S \eta_\Lambda(w_k \mid X_{s'}^e)\right), \tag{30}$$

with ζ_S being a positive constant and $S_k(e)$ being the entire set of possible beginning endpoints of keyword w_k, given the ending endpoint e.

$$\omega_\Lambda^2(w_k \mid X_s^e) = \tilde{1}(\eta_\Lambda(w_k \mid X_s^e) - v_\Lambda(w_k \mid X, e)), \tag{31}$$

where

$$v_\Lambda(w_k \mid X, e) = \frac{1}{\zeta_E} \ln \frac{1}{|E_k(e)|} \sum_{e' \in E_k(e)} \exp(\zeta_E \chi_\Lambda(w_k \mid X, e')), \tag{32}$$

with ζ_E being a positive constant and $E_k(e)$ being the entire set of "near" ending endpoints to e; e.g., $E_k(e) = \{e' \mid e' \in [\, e - \kappa_k, e + \kappa_k \,]\}$ where κ_k is a small natural number constant.

$$\omega_\Lambda^3(w_k \mid X_s^e) = \tilde{1}(\eta_\Lambda(w_k \mid X_s^e) - h_k), \tag{33}$$

where h_k is the decision threshold for keyword w_k. Accordingly, we define the pruning function $\omega_\Lambda(w_k \mid X_s^e)$ as the product of the above three auxiliary functions:

$$\omega_\Lambda(w_k \mid X_s^e) = \omega_\Lambda^1(w_k \mid X_s^e) \cdot \omega_\Lambda^2(w_k \mid X_s^e) \cdot \omega_\Lambda^3(w_k \mid X_s^e). \tag{34}$$

3.3 Simplification for Practical Use

In 2.2, we mentioned the need for reducing the enormous computation of MECK. To make MECK easier and more practical to use, we discuss in this subsection four solutions to simplification.

First, let us recall that the computation of L_p-norm requires a huge amount of calculations, e.g., Eq. (11), and the use of this smooth function was motivated by the attempt to approximate the discontinuous maximum/minimum operations by a smooth function. As in literature on MCE/GPD (e.g., see [7]), by letting the power parameter of the L_p norm function (e.g., ξ in (11)) go to infinity, we can replace in practice this computation by the original simple maximum/minimum search operation without any serious performance drawback. In particular, this type of simplification is effective when the computation including nonlinear time normalization, i.e., path matching, can be replaced by dynamic programming techniques; e.g., see (26).

Second, let us recall that the continuous pruning function using a smoothed step function $\tilde{1}(\cdot)$ does not really prune any keyword hypothesis because this function never becomes zero. Since our concern is mainly on the range around the pruning decision boundary, we can use, with a small compromise in mathematical rigorousness, a piece-

wise smooth step function, e.g.,

$$\tilde{1}(x) = \begin{cases} 0, & \text{if } x \leq -\varsigma \\ \frac{1}{2}\left(1+\frac{x}{\varsigma}\right)^2, & \text{if } -\varsigma < x \leq 0 \\ 1-\frac{1}{2}\left(1-\frac{x}{\varsigma}\right)^2, & \text{if } 0 < x \leq \varsigma \\ 1, & \text{if } \varsigma < x, \end{cases} \tag{35}$$

where ς is a positive constant controlling the smoothness of the function. A sufficiently small ς reduces the number of word hypotheses in the design (training) stage.

Third, any attempt to train all the possible adjustable parameters makes the training convergence slow and often causes the robustness problem that has been widely studied in statistical pattern recognition. Therefore, one solution to this problem can be to tie some parameters, e.g., $h_1 = h_2 = \cdots = h_K$, and to fix some parameters, e.g., make all Σ_{jnm}'s be an identity matrix.

Lastly, we have to simplify the infinite adjustment of (19) which is obviously impractical. Our natural solution to this issue is to use a finite, LVQ-like adjustment [14].

4 Experiment

In this section, we report fundamental performances of our simplified design method by using a real speech corpus and preliminarily selected settings for controllable (selectable) parameters such as the size of subword models and the number of reference vectors.

We conducted evaluation experiments for a task of spotting samples of 10 keyword-classes, i.e., *kaigi*, *kokusai*, *denwa*, *kyouto*, *happyou*, *tsuuyaku*, *nihongo* (*nippongo*), *touroku*, *jimukyoku*, and *ronbun*, from 115 different Japanese sentences, each spoken by 10 female and 10 male speakers. Speech was converted to a 16 Mel-scale power specturm vector every 5 msec. The acoustical feature vector $\boldsymbol{x}_i$ was a concatenation of 7 adjacent power spectra. The sentences were divided into 4 independent sets; Sets 1–3, each consisting of 25 sentences, and Set 4 consisting of 40 sentences. Each Set included at least one sample for every keyword class. Only a (roughly) half set of the sentences of each Set included keyword samples; which means that the remaining sentences included no keyword. Three Sets were used for design and the remaining Set was used for testing; e.g., Set 1–3 for design and Set 4 for testing. The spotting task mode was, therefore, multi-speaker and task-independent.

Each keyword was represented by one or two keyword models, taking into account acoustical variation such as unvocalization; e.g., *nihongo* was represented by two models, "nihongo" and "nippongo",

and *kokusai* was represented by "kokusai" and "koksai". Basically, a subword model was matched to a single phoneme segment. However, some acoustically complex phoneme segments, e.g., "ky" or "ppy", were exceptionally modeled by concatenated subword models. This segmentation was done by minimum-distortion segmentation using the Euclidean distance [15]. All of the subword models were of one-state for all subword classes; $N = 1$. The number of reference vectors (the corresponding covariance matrices) was set to 25 for all subword classes and all states; $M = 25$.

The referene vector set of these subword models was initialized by running k-means clustering over manually selected phoneme segments. The coefficient set $\phi_j = \{\phi_{j0}, \phi_{j1}\}$ was preliminarily initialized using the average μ_j and the standard deviation σ_j of the state distances; $\phi_{j0} = \mu_j/\sigma_j$ and $\phi_{j1} = -1/\sigma_j$. Each matching path, θ, was restricted by the minimum and maximum (emperically selected) duration of each state. The weighting factor $\rho_{\theta l}$ was selected so as to simply average state distances; $\rho_{\theta l} = 1/L_\theta$. The pruning threshold h_k tied over all of the possible keyword classes was initialized to -4.0, which approximately corresponded to the *a posteriori* probability 0.018. The smoothness factor ς of the piece-wise smooth step function was set as follows: 0.5 for $\omega_\Lambda^3(\cdot)$, 0 for $\omega_\Lambda^1(\cdot)$, and 0 for $\omega_\Lambda^2(\cdot)$. Also, $\mathcal{G}$ was set so that the unrealistic overlap between adjacent word hypotheses could be eliminated. Neither grammatical nor semantic constraints were used.

Table 1 summarizes the statistics of our tasks. In the table, for each data set, "# Sentence" and "# Keyword" indicate the total number of sentences and the total number of keywords, respectively.

Table 2 shows the initial testing accuracies of our spotters, each obtained by using only k-means clustering. Note that the accuracies for each set were obtained by a spotter designed using the other three sets of data. In the table, "# Correct-K-S" represents the number of correctly classified keyword-sequences; "# Possible-K-W" represents the number of correct keyword-sequences that remained after pruning the keywords in the spotting stage; and "# Spotted-W" represents the number of keywords that remained after pruning, i.e., keywords actually spotted. The table shows that the spotters spotted an enormous number of keywords, most of which retained the correct keyword-sequences in the beginning of the keyword-sequence classification process, and correctly classified these sequences with an accuracy range of only 5–30%.

After this initialization, we trained the spotters using the method described in Section 2.3. Then, for simplicity and training focusing on spotting errors, we actually defined the loss $\ell(\cdot)$ in (14) using the

following smoothed step function:

$$\tilde{1}(x) = \begin{cases} 0, & \text{if } x < 0 \\ \dfrac{1 - \exp(-x/\varsigma)}{1 + \exp(-x/\varsigma)}, & \text{if } x \geq 0, \end{cases} \tag{36}$$

where $\varsigma = 2.0$. To treat the bounded search space $(-\infty, 0)$ of ϕ_{j1} properly, we adjusted ϕ'_{j1} $(= \ln(-\phi_{j1}))$ instead of ϕ_{j1}. We also fixed, through the overall training, the matrix U in (19) to a diagonal matrix where each diagonal component was 10.0 for the reference vector components; ϕ_{j0} to 0.1; ϕ'_{j1} to 0.02; and h_k to 0.1. Each training sentence was presented six times, and every presentation was set in a random order. We set the learning factor sequence ϵ_t so that $\epsilon_t = \epsilon_0\{1 - (t-1)/T\}$ where $\epsilon_0 = 0.01$ and set T to the total number of iterations, which was a finite approximation of an infinite sequence satisfying the convergence requirement. Concerning the simplification in 2.2, we specially chose two different conditions of simplification: 1) $\gamma = 0.0$ and 2) $\gamma = 1.0 \times 10^{-3}$.

Table 3 shows the accuracies for these two cases. The keyword-sequence classification accuracies were increased to a range of 25–87%, while "# Spotted-W's", i.e., the computation amounts, stayed at a range of 66–103% for $\gamma = 0.0$ and were reduced to a range of 13–29% for $\gamma = 1.0 \times 10^{-3}$. The results clearly proved the high utility of the proposed design method.

5 Summary

We have presented a new spotter design method, called the Minimum Error Classification of Keyword-sequences method (MECK), that is directly linked with continuous speech recognition. The method is characterized by

1. formalizing the spotting-based keyword-sequence classification (continuous speech recognition) in a quite general but vigorous manner,

2. introducing the *a posteriori* odds-based discriminant function that allows one to greatly reduce computation and to easily incorporate a wide range of artificial intelligence techniques such as fuzzy logic in speech recognition,

3. making possible a spotter design that does not use labeled design samples, which frees one from the troublesome labeling work that is necessary in conventional design methods,

4. incorporating keyword hypothesis pruning process for unified loss minimization.

Experiments using a prototype-based spotter in Japanese keyword spotting tasks clearly demonstrated the marked utility of our design method.

Note that the design method proposed in this paper is available for problems other than spotting-based speech recognition. Our method can be applied to any inference problem being formalized as a decision combination problem if log *a posteriori* odds estimator functions and pruning functions are addequately chosen.

References

[1] W. Ward, "Understanding Spontaneous Speech: The Phoenix System," IEEE, Proc. of ICASSP-91, S5.29, pp. 365–367 (1991).

[2] H. Tsuboi and Y. Takebayashi, "A Real-Time Task-Oriented Speech Understanding System Using Keyword-Spotting," IEEE, Proc. of ICASSP-92, Vol. 1, pp. 197–200 (1992).

[3] J. G. Wilpon, L. R. Rabiner, C.-H. Lee, and E. R. Goldman, "Automatic Recognition of Keywords in Unconstrained Speech Using Hidden Markov Models," IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. 38, No. 11, pp. 1870–1878 (1990).

[4] J. R. Rohlicek, W. Russell, S. Roukos, and H. Gish, "Continuous Hidden Markov Modelling for Speaker-Independent Word Spotting," IEEE, Proc. of ICASSP-89, S12.8, pp. 627–630 (1989).

[5] L. T. Niles, L. D. Wilcox, and M. A. Bush, "Error-Correcting Training for Phoneme Spotting," IEEE, Proc. of ICASSP-92, Vol. 1, pp. 425–428 (1992).

[6] R. C. Rose, "Discriminant Wordspotting Techniques for Rejecting Non-Vocabulary Utterances in Unconstrained Speech," IEEE, Proc. of ICASSP-92, Vol. 2, pp. 105–108 (1992).

[7] T. Komori and S. Katagiri, "A New Learning Algorithm for Minimizing Spotting Errors," Proc. of 1993 IEEE Workshop on Neural Networks for Signal Processing, pp. 333–342 (1993).

[8] T. M. English and L. C. Boggess, "Back-Propagation Training of a Neural Network for Word Spotting," IEEE, Proc. of ICASSP-92, Vol. 2, pp. 357–360 (1992).

[9] T. Zeppenfeld and A. H. Waibel, "A Hybrid Neural Network, Dynamic Programming Word Spotter," IEEE, Proc. of ICASSP-92, Vol. 2, pp. 77–80 (1992).

[10] Y. Takebayashi, H. Tsuboi, and H. Kanazawa, "Keyword-Spotting in Noisy Continuous Speech Using Word Pattern Vector

Subabstraction and Noise Immunity Learning," IEEE, Proc. of ICASSP-92, Vol. 2, pp. 85–88 (1992).

[11] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis* (Wiley, New York, 1973).

[12] B.-H. Juang and S. Katagiri, "Discriminative Learning for Minimum Error Classification," IEEE Trans. on Signal Processing, Vol. 40, No. 12, pp. 3043–3054 (1992).

[13] S. Katagiri, C.-H. Lee, and B.-H. Juang, "New Discriminative Training Algorithms Based on the Generalized Probabilistic Descent Method," Proc. of the 1991 IEEE Workshop on Neural Networks for Signal Processing, pp.299–308 (1991).

[14] E. McDermott and S. Katagiri, "LVQ-Based Shift-Tolerant Phoneme Recognition," IEEE Trans. on Signal Processing, Vol. 39, No. 6, pp. 1398–1411 (1991).

[15] T. Svendsen and F. Soong, "On the Automatic Segmentation of Speech Signals," AT&T Bell Labs, Tech. Memo. No. 11226-870416-10 (1987).

Table 1: Characteristic of Speech Corpus

	Set 1	Set 2	Set 3	Set 4	Total
# Sentence	500	500	500	800	2300
# Keyword	480	620	1320	300	2720

Table 2: Initial Spotter Performance

	Set 1	Set 2	Set 3	Set 4	Total
# Correct-K-S	58	93	23	250	424
# Possible-K-S	491	489	476	791	2247
# Spotted-W	20241	15612	23147	17932	76932

Table 3: Spotter Accuracies after MECK Training

a) $\gamma = 0.0$

	Set 1	Set 2	Set 3	Set 4	Total
# Correct-K-S	379	361	152	695	1587
# Possible-K-S	500	495	365	800	2160
# Spotted-W	20884	13624	15184	15226	64918

b) $\gamma = 1.0 \times 10^{-3}$

	Set 1	Set 2	Set 3	Set 4	Total
# Correct-K-S	257	273	127	701	1358
# Possible-K-S	356	400	236	797	1789
# Spotted-W	2540	2426	5067	5212	15245