Computer Speech and Language (2001) **00**, 1–34
Article No. 10.1006/csla.2001.0188
Available online at http://www.idealibrary.com on IDEAL®

Theory and practice of acoustic confusability

Harry Printz†§ and Peder A. Olsen‡¶

†Agile TV Corporation, 333 Ravenswood Ave, Bldg 202, Menlo Park, CA 94025, U.S.A., ‡IBM T J Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, U.S.A.

Abstract

In this paper we define two alternatives to the familiar perplexity statistic (hereafter lexical perplexity), which is widely applied both as a figure of merit and as an objective function for training language models. These alternatives, respectively *acoustic perplexity* and the *synthetic acoustic word error rate*, fuse information from both the language model and the acoustic model. We show how to compute these statistics by effectively synthesizing a large acoustic corpus, demonstrate their superiority (on a modest collection of models and test sets) to lexical perplexity as predictors of language model performance, and investigate their use as objective functions for training language models. We develop an efficient algorithm for training such models, and present results from a simple speech recognition experiment, in which we achieved a small reduction in word error rate by interpolating a language model trained by synthetic acoustic word error rate with a unigram model.

Publisher:
Please supply received and accepted dates

1. Introduction

Let P_θ be a language model, where $P_\theta(w_0 \dots w_{S-1})$ is the probability that the model assigns to word sequence $w_0 \dots w_{S-1}$, and θ is a (typically very large) set of parameters, which determine the numerical value of this probability. One widely-used method of assigning values to the elements of θ is to obtain a corpus $\mathcal{C} = w_0 \dots w_{N-1}$ of naturally generated text, also very large, and to adjust θ to maximize $P_\theta(\mathcal{C})$, the modeled probability of the corpus. This is an instance of the well-established principle of maximum-likelihood estimation: the model is made to accord as closely as possible with a collection of examples, in the hope that it will function well as a predictor in the future.

Since the aim of modeling is to assign high probability to events that are known to occur, while assigning little or none to those that do not, this approach is eminently reasonable. Because the familiar quantity

$$Y_L(P_\theta, \mathcal{C}) = P_\theta(\mathcal{C})^{-1/|\mathcal{C}|}, \tag{1}$$

called the perplexity (Bahl, Baker, Jelinek & Mercer, 1977), is inversely related to the raw likelihood $P_\theta(\mathcal{C})$, the assumption has likewise arisen that the lower the perplexity the better.

§E-mail: printz@agile.tv
¶E-mail: pederao@us.ibm.com

0885–2308/01/000001 + 34 $35.00/0

Perplexity is commonly reported in papers on language modeling as evidence of the value of the author's new insight or mathematical technique.

There is, of course, nothing wrong with this point of view, as far as it goes. If the goal is to predict the next word of text w_i given a history $h_i = w_0 \dots w_{i-1}$ of preceding words—say for a text compression system—it is clear that perplexity (or, equivalently, likelihood) is an appropriate figure of merit. But when a language model functions as a component of a statistical speech recognition (Bahl, Jelinek & Mercer, 1983) or machine translation (Berger *et al.*, 1994) system, perplexity may not be a good predictor of accuracy. The following excerpt from a recent paper by two experienced language modelers drives this point home:

> This paper has described two simple adaptive language models, and shown that while they lead to substantial reductions in perplexity over the baseline Broadcast News language model, they do not result in improved recognition performance. . . . [T]hese results, as well as those given in several other papers, show that even fairly large reductions in perplexity are no guarantee of a reduction in word error rate (Clarkson & Robinson, 1998).

The inadequacy of perplexity is widely acknowledged, and the search for a substitute is the subject of previous research (Chen, Beeferman & Rosenfeld, 1998; Ferretti, Maltese & Scarci, 1990; Ito, Kohda & Ostendorf, 1999; Iyer, Ostendorf & Meteer, 1997).

In this paper, we investigate a statistic, called *acoustic perplexity*, which we propose to use both as a measure-of-goodness for evaluating language models, and as a computational principle for constructing them. Acoustic perplexity differs fundamentally from most other proposed measures, in that it incorporates the characteristics of the acoustic channel. Specifically, acoustic perplexity, and the related quantity *synthetic acoustic word error rate*, which we also define later, provide a better indication of how well a language model will function when used as a component of a speech recognition system.

In our development, we show how acoustic perplexity is a natural extension of the existing notion of perplexity—hereafter called *lexical perplexity*, when confusion with acoustic perplexity may arise. We analyze this notion mathematically, and define it in terms of another quantity we introduce, the *acoustic encoding probability*. This in turn can be understood through a still more fundamental expression, the *acoustic confusability* of lexeme pairs. By manipulating the hidden Markov models that are standard in automatic speech recognition, we develop rigorous and intuitively appealing computational methods for determining all of these quantities.

Moreover, we provide experimental evidence to substantiate our claim that acoustic perplexity, and the synthetic acoustic word error rate (hereafter SAWER), are better measures of language model quality than lexical perplexity. We supply an algorithm for training a language model by minimization of the SAWER, and give experimental results for a model so trained. The use of the SAWER language model does not by itself yield higher accuracy than a model trained to minimize perplexity, but we demonstrate a small gain when the two are combined. Finally, we show how to apply our computational methods to such practical problems as choosing maximum entropy features for statistical language modeling, and selecting the vocabulary for a speech recognition system.

These methods are sufficiently general that they may be applied to any source-channel decoding paradigm where the source is explicitly modeled. For instance, statistical machine translation is another appealing application of this technique.

2. Motivating numerical example

In this section we present an extended numerical example, based upon the operation of a statistical speech recognition system, that demonstrates a fundamental weakness of lexical perplexity: it is possible to *lower* the lexical perplexity, even when measured with respect to the very text that is being decoded, and yet *raise* the error rate on this same text. Our example greatly simplifies and idealizes the operation of such a system, and the numbers we use are contrived to prove our point. Nevertheless it is plausible. We present it to introduce the key concepts we will be manipulating, and to motivate and develop intuition about acoustic perplexity. It should be noted, however, that when the training and test corpora have identical statistics—as they do in the example—the maximum-likelihood model in fact yields the lowest expected error rate.

2.1. Preliminaries

To commence we recall the source-channel paradigm for statistical speech recognition (Bahl *et al.*, 1983). Let $P(\mathcal{S})$ model the text source, and let $P(\mathcal{A} \mid \mathcal{S})$ model the text-to-acoustic channel that comprises the human speech apparatus. Here $\mathcal{S}$ stands for some text, and $\mathcal{A}$ stands for an acoustic signal. By Bayes' theorem $P(\mathcal{S} \mid \mathcal{A}) = P(\mathcal{A} \mid \mathcal{S})P(\mathcal{S})/P(\mathcal{A})$; hence we may recover the most likely source text $\hat{\mathcal{S}}$ for fixed acoustics $\mathcal{A}$ via

$$\hat{\mathcal{S}} = \arg\max_{\mathcal{S}} P(\mathcal{S} \mid \mathcal{A}) = \arg\max_{\mathcal{S}} \frac{P(\mathcal{A} \mid \mathcal{S})P(\mathcal{S})}{P(\mathcal{A})} = \arg\max_{\mathcal{S}} P(\mathcal{A} \mid \mathcal{S})P(\mathcal{S}). \tag{2}$$

The right-most equality is justified on the grounds that $P(\mathcal{A})$ is constant with respect to variations in $\mathcal{S}$.

For concreteness, let us now suppose that we are decoding the utterance $\mathcal{A} = a(\textit{she})a(\textit{skis})$ $a(\textit{well})$, and that these are the only three words in the vocabulary, hereafter abbreviated *s*, *k*, *w*, respectively. The notation $a(\ldots)$ indicates that we are decoding an *acoustic event* corresponding to the given word, which is to say, an audio signal.

We make three additional simplifying assumptions. First, the language models in this example will all be unigram models, and thus

$$P(\mathcal{S}) = P(w_0 \ldots w_{N-1}) = p(w_0) \cdots p(w_{N-1}) \tag{3}$$

for any word sequence $\mathcal{S} = w_0 \ldots w_{N-1}$. Second, the channel model exhibits perfect conditional independence. By this we mean for any three-word sequence $x\ y\ z$, we have

$$P(a(s)\ a(k)\ a(w) \mid x\ y\ z) = p(a(s) \mid x)\ p(a(k) \mid y)\ p(a(w) \mid z). \tag{4}$$

Each of the factors on the right-hand side is an *acoustic encoding probability*. That is, $p(a(s) \mid x)$ is the probability that the acoustic event $a(s)$ will be observed, given that the word x was spoken. Third, we will suppose that in addition to restricting our vocabulary to just three words, the acoustic space consists of the finite space $\Omega = \{a(s), a(k), a(w)\}$.

We proceed to set up some numerical values for our example. Note that there are three distributions to define, namely $p(a(\cdot) \mid s)$, $p(a(\cdot) \mid k)$, and $p(a(\cdot) \mid w)$. For the purposes of this example, we impose two stringent conditions upon these distributions. The first is that

$$p(a(x) \mid x) \geq p(a(y) \mid x) \quad \forall x,\ y \in V. \tag{5}$$

(Here V is the vocabulary; in this case $V = \{s, k, w\}$.) We will say that such a distribution $p(a(\cdot) \mid x)$ is *generatively faithful to* x, or simply that it *generates faithfully*. The second condition

$$p(a(x) \mid x) \geq p(a(x) \mid y) \quad \forall x, y \in V, \tag{6}$$

states that the likelihood of acoustic event $a(x)$ given x is at least as large as its likelihood given y. We will say that such a set of likelihoods *decodes faithfully*.

Here is a set of numerical values that both generates faithfully and decodes faithfully:

$$\begin{array}{lll} p(a(s) \mid s) = \frac{2}{3} & p(a(k) \mid s) = \frac{1}{3} & p(a(w) \mid s) = 0 \\ p(a(s) \mid k) = \frac{5}{12} & p(a(k) \mid k) = \frac{7}{12} & p(a(w) \mid k) = 0 \\ p(a(s) \mid w) = 0 & p(a(k) \mid w) = 0 & p(a(w) \mid w) = 1. \end{array}$$

Note that each row is a distribution $p(a(\cdot) \mid x)$ and therefore sums to unity, and that each column is a set of likelihoods $p(a(x) \mid \cdot)$ and need not sum to unity. The generative condition is a requirement on each row, and the decoding condition is a requirement on each column. For both conditions, for the values given, the inequalities are satisfied strongly throughout. Moreover, these values reflect our intuitions about the confusabilities of the words *she, skis, well*. Since all probabilities and likelihoods for *well* are 0, except for $p(a(w) \mid w)$ which then necessarily equals 1, we say that this word has *infinitely sharp acoustics*.

2.2. Decoding example

We proceed to demonstrate the promised counterintuitive result: it is possible to decrease language model perplexity, as measured on a test corpus, and yet increase the word error rate on this same corpus. This is so even under the assumptions that the acoustic encoding probabilities both generate and decode faithfully.

Consider two language models, P and P', determined, respectively, by the following tables of values.

$$\begin{array}{llll} \text{model } P & p(s) = \frac{3}{8} & p(k) = \frac{1}{2} & p(w) = \frac{1}{8} \\ \text{model } P' & p'(s) = \frac{1}{4} & p'(k) = \frac{1}{2} & p'(w) = \frac{1}{4}. \end{array}$$

From these figures, we calculate the likelihoods and the perplexities of both models on a test corpus $\mathcal{T}$, consisting of the single sentence *she skis well*.

$$\begin{array}{ll} P(\mathcal{T}) = \frac{3}{128} & Y_L(P, \mathcal{T}) = \sqrt[3]{\frac{128}{3}} \approx 3{\cdot}49 \\ P'(\mathcal{T}) = \frac{4}{128} & Y_L(P', \mathcal{T}) = \sqrt[3]{\frac{128}{4}} \approx 3{\cdot}17. \end{array}$$

It is clear that model P' has higher likelihood on $\mathcal{T}$ than model P, and hence lower perplexity. By the maximum likelihood principle, we would assume that P' is a better model of the text that we are decoding than P.

Now we observe the effect of decoding with models P and P', respectively. That is, for the given acoustics, we solve Equation (2) explicitly for each language model. Consider model P first. Writing $x\ y\ z$ for our guesses of the decoded words in order, we have

$$\hat{\mathcal{S}} = \arg\max_{\mathcal{S}}\ P(\mathcal{A} \mid \mathcal{S})\ P(\mathcal{S}) \tag{7}$$

$$= \arg\max_{xyz}\ P(a(s)\ a(k)\ a(w) \mid x\ y\ z)\ P(x\ y\ z) \tag{8}$$

$$= \arg\max_{x} p(a(s)|x)p(x),\ \arg\max_{y} p(a(k)|y)p(y),\ \arg\max_{z} p(a(w)|z)p(z). \tag{9}$$

In the last line we reordered the factors to obtain three products that depend exclusively upon x, y and z, respectively. This allows the maximizations to proceed independently. Thus to determine the decoding of acoustics $a(s)$ with language model P, we need only compute the product $p(a(s)|x)p(x)$ for the three values $x = s,\ k,\ w$; the choice of x that yields

the maximum is our decoder's output. We do likewise for $a(k)$ and $a(w)$. This computation is carried out in the table below, for the full utterance $\mathcal{A} = a(\textit{she})\ a(\textit{skis})\ a(\textit{well})$. The maximum for each column—corresponding to the decoded word—is enclosed in a box.

$$\begin{array}{lll}
p(a(s)|s)\ p(s) = \boxed{\frac{1}{4}} & p(a(k)|s)\ p(s) = \frac{1}{8} & p(a(w)|s)\ p(s) = 0 \\
p(a(s)|k)\ p(k) = \frac{5}{24} & p(a(k)|k)\ p(k) = \boxed{\frac{7}{24}} & p(a(w)|k)\ p(k) = 0 \\
p(a(s)|w)\ p(w) = 0 & p(a(k)|w)\ p(w) = 0 & p(a(w)|w)\ p(w) = \boxed{\frac{1}{8}}.
\end{array}$$

Since the boxed choices correspond to *she*, *skis*, *well*, respectively, language model P yields an error-free decoding.

Next we decode with the identical acoustic model, but using P' as the language model. As before, we box the maximum for each acoustic segment.

$$\begin{array}{lll}
p(a(s)|s)\ p'(s) = \frac{1}{6} & p(a(k)|s)\ p'(s) = \frac{1}{12} & p(a(w)|s)\ p'(s) = 0 \\
p(a(s)|k)\ p'(k) = \boxed{\frac{5}{24}} & p(a(k)|k)\ p'(k) = \boxed{\frac{7}{24}} & p(a(w)|k)\ p'(k) = 0 \\
p(a(s)|w)\ p'(w) = 0 & p(a(k)|w)\ p'(w) = 0 & p(a(w)|w)\ p'(w) = \boxed{\frac{1}{4}}.
\end{array}$$

The acoustics for the first word are $a(\textit{she})$, but we have decoded *skis* in this position: language model P' yields a decoding error.

2.3. Analysis

We have a conundrum. The lexical perplexity as measured on a test corpus has gone down, yet the error rate on this same corpus has gone up. Moreover, this cannot be explained as a search error, or a consequence of finite-precision arithmetic. Our decoding procedure has searched exhaustively, and our computations are exact.

The problem is that lexical perplexity is too coarse a measure of goodness. First let us see why its value has gone down. Since each word of the vocabulary occurs exactly once in $\mathcal{T}$, the best model of $\mathcal{T}$, in the maximum-likelihood sense, is of course the uniform model. [This follows immediately from the Gibbs inequality, $\sum_i p_i \log q_i \leq \sum_i p_i \log p_i$ where $\langle p_i \rangle$ and $\langle q_i \rangle$ are probability distributions over the same discrete space (Cover & Thomas, 1991, Theorem 2.6.3).] The model P' gives equal probabilities for s and w; in model P these values differ. Holding the probability of k constant, this adjustment between s and w naturally improves the likelihood.

But this greater smoothness is achieved by robbing probability mass from s to give to w, when w's acoustics alone are enough to decode the word cleanly. This is so no matter what probability the language model accords to w, providing it is non-zero. Somehow we want to tell our training procedure—if you are interested in a smoother language model, do what you can to equate the probabilities of s and k, we do not really care about w! But it is only because we have inspected the $p(a(w) \mid \cdot)$ that we know $p(w)$ does not matter. The lexical perplexity is completely insensitive to the channel probabilities.

3. Acoustic perplexity

3.1. Definition of acoustic perplexity

We proceed to motivate and define the acoustic perplexity. Later we will relate the methods presented here to the synthetic acoustic word error rate.

The heart of our argument is the observation, presented in the preceding section, that training the language model by

$$\hat{\theta} = \arg\max_{\theta} P_\theta(\mathcal{C}) \tag{10}$$

can be misleading. This is not because there is something wrong with lexical perplexity. Rather, it is the wrong objective function for the task at hand. Surely, if our aim is to train our model so we decode text $\mathcal{C}$ from its acoustic realization $\mathcal{A}$, we should adjust our model's parameters θ according to

$$\hat{\theta} = \arg\max_{\theta} P_\theta(\mathcal{C} \mid \mathcal{A}) \tag{11}$$

where $P_\theta(\mathcal{C} \mid \mathcal{A})$ is the reverse channel model constructed and manipulated as in Equation (2). That is, since $P_\theta(\mathcal{C} \mid \mathcal{A})$ is the model used to decode, it seems natural to us to adopt it as the objective function of our language model training procedure.

Note that while training by lexical perplexity requires only a large text corpus $\mathcal{C}$, training by acoustic perplexity requires both text $\mathcal{C}$ and acoustics $\mathcal{A}$, where the latter is a spoken version of the former. We will refer to the pair $\langle \mathcal{C}, \mathcal{A} \rangle$ as a *joint corpus*. The need for a large joint corpus is a practical obstacle that we address later in this paper.

It might be argued, along the lines of Equation (2), that we are already maximizing $P_\theta(\mathcal{C} \mid \mathcal{A})$, since

$$\arg\max_{\theta} P_\theta(\mathcal{C} \mid \mathcal{A}) = \arg\max_{\theta} (P(\mathcal{A} \mid \mathcal{C}) / P(\mathcal{A})) \cdot P_\theta(\mathcal{C}) = \arg\max_{\theta} P_\theta(\mathcal{C}). \tag{12}$$

The fallacy in this argument lies in assuming $P(\mathcal{A})$ is constant with respect to θ. In applying Bayes' theorem, the denominator must be the marginal $P_\theta(\mathcal{A})$, defined as

$$P_\theta(\mathcal{A}) = P(\mathcal{A} \mid \mathcal{C}_1) P_\theta(\mathcal{C}_1) + P(\mathcal{A} \mid \mathcal{C}_2) P_\theta(\mathcal{C}_2) + \cdots. \tag{13}$$

And though $\mathcal{A}$ is fixed—as is $\mathcal{C}$ itself—for any particularly training instance, the quantity $P_\theta(\mathcal{A})$ of course varies with θ, which are the quantities being adjusted. Thus the right-most equality in (12) is false.

We return to the main line of development. As an exact analog of lexical perplexity, we define the quantity

$$Y_A(P_\theta, \mathcal{C}, \mathcal{A}) = P_\theta(\mathcal{C} \mid \mathcal{A})^{-1/|\mathcal{C}|}, \tag{14}$$

the *acoustic perplexity* of the model P_θ, evaluated on lexical corpus $\mathcal{C}$ and its acoustic realization $\mathcal{A}$. Moreover, in the same way as P_θ is decomposed into a product of individual-word probabilities, for use in computing Y_L, so too may $P_\theta(\mathcal{C} \mid \mathcal{A})$ be decomposed.

To express this decomposition, we adopt the following notational conventions. The word we are decoding, at the current position i of the corpus, is w_i. Its acoustic realization is written $a(w_i)$. The sequence of all words preceding w_i, which is $w_0\ w_1 \ldots w_{i-1}$, is denoted h_i; its acoustic realization is $a(h_i)$. Likewise the sequence of all words following w_i is written r_i, with acoustics denoted $a(r_i)$. (Here the letter r is used to suggest right context.) The complete situation is summarized in Figure 1. By elementary probability theory (Chung, 1979, Section 5.2, Proposition 1), we have the familiar decomposition

$$P_\theta(\mathcal{C}) = P_\theta(w_0\ w_1 \ldots w_{|\mathcal{C}|-1}) = \prod_{i \in \mathcal{C}} p_\theta(w_i \mid h_i). \tag{15}$$

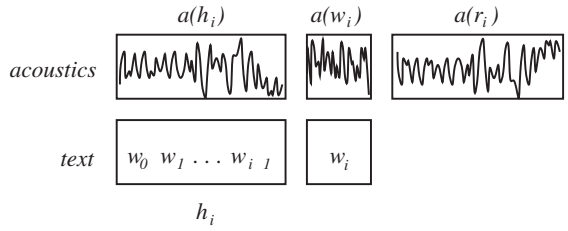

Figure 1. Notational conventions for text and acoustics. w_i is the current word, h_i is the (textual) history, and $a(h_i)$, $a(w_i)$ and $a(r_i)$ are the acoustics of the history, the current word, and the right context, respectively.

We identify P_θ with the family of conditional distributions $\{p_\theta(w \mid h)\}$ that underlies it, and speak of them as "a language model". We say a language model is *smooth* if for all w and h, we have $p(w \mid h) > 0$.

By the rules of conditional probability we have

$$P_\theta(\mathcal{C} \mid \mathcal{A}) = \prod_{i \in \mathcal{C}} p_\theta(w_i \mid h_i \; a(w_0 \; w_1 \ldots w_{|\mathcal{C}|-1})) \tag{16}$$

$$= \prod_{i \in \mathcal{C}} p_\theta(w_i \mid h_i \; a(h_i \; w_i \; r_i)), \tag{17}$$

where the second line is a purely notational variant of the first. This expression is appropriate for recognition of continuous speech. For recognition of discrete speech, where each word is unmistakably surrounded by silence, we have the simpler form

$$P_\theta(\mathcal{C} \mid \mathcal{A}) = \prod_{i \in \mathcal{C}} p_\theta(w_i \mid h_i \; a(w_i)). \tag{18}$$

Next we show how the language model probability enters explicitly into this expression. Consider any one factor in (17). Suppressing the i subscript for readability, by Bayes' theorem we may write

$$p_\theta(w \mid h \; a(h \; w \; r)) = \frac{p(a(h \; w \; r) \mid w \; h) \; p_\theta(w \mid h)}{\sum_{x \in V} p(a(h \; w \; r) \mid x \; h) \; p_\theta(x \mid h)}. \tag{19}$$

Here $p_\theta(w \mid h)$ and $p_\theta(x \mid h)$ are regular language model probabilities. For the case of discrete speech, this expression takes the form

$$p_\theta(w \mid h \; a(w)) = \frac{p(a(w) \mid w \; h) \; p_\theta(w \mid h)}{\sum_{x \in V} p(a(w) \mid x \; h) \; p_\theta(x \mid h)}. \tag{20}$$

We will refer to the family of conditional distributions $\{p(a(h \; w \; r) \mid x \; h)\}$, or just $\{p(a(w) \mid xh)\}$ for the discrete case, as an *acoustic encoding model*.

3.2. Limiting behavior of acoustic perplexity

We now consider two special limiting cases, respectively infinitely sharp acoustics and perfectly smooth acoustics. We will show in the former case, all language models are equivalent, and in the latter, acoustic perplexity reduces to the familiar notion of lexical perplexity.

By *infinitely sharp acoustics*, we mean that for arbitrary history h and right context r, the acoustic encoding model is given as

$$p(a(h \; w \; r) \mid h \; x) = \eta_{hr} \cdot \delta(w, x) \tag{21}$$

for an appropriate normalizer η_{hr}. A language model P_θ is smooth if every underlying conditional probability $p_\theta(x \mid h)$ is non-zero.

Theorem 3.1: *Consider joint corpus $\langle \mathcal{C}, \mathcal{A} \rangle$, and let P_θ be a smooth language model. If the acoustic encoding model is infinitely sharp, then $Y_A(P_\theta, \mathcal{C}, \mathcal{A}) = 1$ for all θ.*

Proof: Consider position i of joint corpus $\langle \mathcal{C}, \mathcal{A} \rangle$. By Bayes' theorem as applied in Equation (19) we have

$$p_\theta(w_i \mid a(h_i\ w_i\ r_i)\ h_i) = \frac{p(a(h_i\ w_i\ r_i) \mid w_i\ h_i) \cdot p_\theta(w_i \mid h_i)}{\sum_{x \in V} p(a(h_i\ w_i\ r_i) \mid x\ h_i) \cdot p_\theta(x \mid h_i)} \tag{22}$$

$$= \frac{\eta_{hr} \cdot \delta(w_i, w_i) \cdot p_\theta(w_i \mid h_i)}{\eta_{hr} \cdot \sum_{x \in V} \delta(w_i, x) \cdot p_\theta(x \mid h_i)} \tag{23}$$

$$= \frac{p_\theta(w_i \mid h_i)}{p_\theta(w_i \mid h_i)} \tag{24}$$

$$= 1. \tag{25}$$

Thus $Y_A(P_\theta, \mathcal{C}, \mathcal{A}) = \left(\prod_i p_\theta(w_i \mid a(h_i\ w_i\ r_i)\ h_i) \right)^{-1/|\mathcal{C}|} = 1$. □

This computation captures the intuition that the sharper the acoustics—that is, the more distinct and unequivocal the pronunciations of the words in the recognizer vocabulary—the less important the language model. In the extreme, if every word sounded like itself alone, the language model would be irrelevant.

Now we consider the opposite extreme. We will say an acoustic encoding model is *perfectly smooth* if

$$p(a(h\ w\ r) \mid x\ h) = \eta_{hr} \cdot \frac{1}{|V|}, \tag{26}$$

where η_{hr} is an appropriate normalizer, and $|V|$ is the vocabulary size.

Theorem 3.2: *Consider joint corpus $\langle \mathcal{C}, \mathcal{A} \rangle$, and let P_θ be a smooth language model. If the acoustic encoding model is perfectly smooth, then $Y_A(P_\theta, \mathcal{C}, \mathcal{A}) = Y_L(P_\theta, \mathcal{C})$. That is, the acoustic perplexity and lexical perplexity are equal.*

Proof: As before, we consider an arbitrary position i of the corpus pair $\langle \mathcal{C}, \mathcal{A} \rangle$. By Bayes' theorem as applied in Equation (19) we have

$$p_\theta(w_i \mid a(h_i\ w_i\ r_i)\ h_i) = \frac{p(a(h_i\ w_i\ r_i) \mid w_i\ h_i) \cdot p_\theta(w_i \mid h_i)}{\sum_{x \in V} p(a(h_i\ w_i\ r_i) \mid x\ h_i) \cdot p_\theta(x \mid h_i)} \tag{27}$$

$$= \frac{(\eta_{hr}/|V|) \cdot p_\theta(w_i \mid h_i)}{(\eta_{hr}/|V|) \cdot \sum_{x \in V} p_\theta(x \mid h_i)} \tag{28}$$

$$= p_\theta(w_i \mid h_i). \tag{29}$$

Thus $Y_A(P_\theta, \mathcal{C}, \mathcal{A}) = \left(\prod_i p_\theta(w_i \mid a(h_i\ w_i\ r_i) h_i) \right)^{-1/|\mathcal{C}|} = \left(\prod_i p_\theta(w_i \mid h_i) \right)^{-1/|\mathcal{C}|} = Y_L(P_\theta, \mathcal{C})$. □

The importance of this result is that it provides us with some insight into the relation between perplexity and the sharpness of the acoustic model. In trying to drive down perplexity we are minimizing acoustic perplexity under the assumption that *all words sound the same*. This assumption is unrealistic. If we believe that acoustic perplexity expresses the performance of speech recognition systems, this explains the sometimes-disappointing recognition results that accompany even major reductions of lexical perplexity. We suspect

that these reductions come from boosting the language model probability of words that the recognizer already decodes cleanly.

3.3. Acoustic perplexity of the example

We are now in a position to make a preliminary, if purely synthetic, test of acoustic perplexity as a measure of goodness. Earlier we saw that language model P' has lower lexical perplexity on test corpus $\mathcal{T}$ than language model P, that is $Y_L(P, \mathcal{T}) > Y_L(P', \mathcal{T})$, but yields a higher word error rate. We proceed to compute the acoustic perplexities of P and P' on the joint corpus $\langle \mathcal{C}, \mathcal{A} \rangle$; we would hope to find $Y_A(P, \mathcal{T}, \mathcal{A}) < Y_A(P', \mathcal{T}, \mathcal{A})$.

Our starting point is Equation (18). For a unigram language model the history h is irrelevant, so in the case of the example's three-word corpus we have

$$Y_A(P, \mathcal{T}, \mathcal{A}) = P(\mathcal{T} \mid \mathcal{A})^{-1/3} = (p(s \mid a(s)) \cdot p(k \mid a(k)) \cdot p(w \mid a(w)))^{-1/3}. \qquad (30)$$

To determine the value of each factor, we use (20), yielding

$$Y_A(P, \mathcal{T}, \mathcal{A}) = \sqrt[3]{\frac{55}{21}} \approx 1{\cdot}38 \qquad \text{and} \qquad Y_A(P', \mathcal{T}, \mathcal{A}) = \sqrt[3]{\frac{81}{28}} \approx 1{\cdot}42 \qquad (31)$$

for the two language models.

Thus we have established $Y_A(P, \mathcal{T}, \mathcal{A}) < Y_A(P', \mathcal{T}, \mathcal{A})$. That is, for these two particular language models, acoustic perplexity and word error rate move in the same direction.

In fact a much stronger result holds. It can be shown that for this example, any adjustment of the language model that reduces the acoustic perplexity, as measured on the test corpus $\mathcal{T}$, must yield either the same or lower word error rate. Indeed, this is so no matter what specific probabilities are assigned to $p(a(s) \mid s)$, $p(a(s) \mid k)$, $p(a(k) \mid s)$ and $p(a(k) \mid k)$, providing these models generate faithfully, and providing we continue to require infinitely sharp acoustics for word w. However, this result holds for this three-word example only and does not generalize to larger corpora.

4. Computational methods

Thus far we have made a case for training a language model by maximization of $P_\theta(\mathcal{C} \mid \mathcal{A})$, rather than maximization of $P_\theta(\mathcal{C})$. We now come to grips with two key obstacles to carrying out this program. First, we must devise a scheme for computing the all-important acoustic encoding probabilities, $p(a(h\ w\ r) \mid h\ x)$, for arbitrary h, w, r and x. Second, we need a way to cope with the lack of a large joint corpus $\langle \mathcal{C},\ \mathcal{A} \rangle$. A typical language model training corpus may contain on the order of one billion (10^9) words of text. But the largest joint corpus that we know of contains a measly three million (3×10^6) or so words of pronounced text. While this may (or may not) suffice to train an acoustic model for a speech recognition system, it surely is not enough to train a language model.

These two obstacles, seemingly unrelated, have a single solution that dovetails them neatly together. The solution is to synthesize the information we would obtain from the desired large joint corpus. To do so we proceed in two steps. First we use our existing (relatively small) joint corpus to build acoustic models; this is just acoustic training as we presently understand it. Then we train the language model on a full-sized textual corpus, using synthetic approximations to the required acoustic encoding probabilities. These synthetic approximations can be computed analytically from the just-trained acoustic models via a technique that we will explain.

This section develops in detail our method for synthetic computation of acoustic encoding probabilities. For simplicity, we will couch the discussion in terms of an isolated-word speech recognition system, and further assume that each word has only one pronunciation. We proceed as follows. First, in Section 4.1, we introduce some nomenclature, at the same time laying the foundation for handling multiple pronunciations. In Section 4.2, we define an acoustic event $a(w)$, and show how to replace true events with models of them. Then in Section 4.3 we explain our scheme for computing acoustic confusabilities. We show how the hidden Markov models commonly used in recognition systems entail summing a doubly-infinite expression, but then give an explicit algorithm for computing this sum in closed form. Finally, in Section 4.4, we discharge the assumption that each word has only one pronunciation, and extend the method to continuous speech.

4.1. Multiple pronunciations

Our aim is to provide a working definition of an acoustic encoding probability, in the discrete case written $p(a(w) \mid h\ x)$. We begin by addressing the issue of multiple pronunciations. For simplicity we ignore h for the moment and consider just $p(a(w) \mid x)$. Here $a(w)$ is an acoustic event and the word x is just a placeholder, to determine which model to use when computing $p(a(w) \mid x)$.

If there were only one single model for x, then $p(a(w) \mid x)$ would be the probability that this model assigns to the observation $a(w)$. But in general a given word x has many pronunciations. We will refer to each one as a *lexeme* and write

$$x = \{l^1(x), l^2(x), \ldots, l^{n_x}(x)\}, \tag{32}$$

where n_x is the number of distinct pronunciations we recognize for x. Carrying this notation a little further, we will write $l(x) \in x$ for an arbitrary lexeme $l(x)$ associated with the word x, and $\sum_{l(x)\in x}$ for a sum in which $l(x)$ varies over the lexeme set for x. We will formalize this notion, and write B for the finite, discrete set of lexemes present in our recognition system, and identify a word $x \in V$ with a subset of B.

By use of standard results from probability theory we have

$$p(a(w) \mid x) = \sum_{l(x)\in x} p(a(w) \mid l(x)) \cdot p(l(x) \mid x). \tag{33}$$

By formal manipulations this extends to arbitrary conditioning h, and so we have

$$p(a(w) \mid x\ h) = \sum_{l(x)\in x} p(a(w) \mid l(x)\ h) \cdot p(l(x) \mid x\ h). \tag{34}$$

From this point on we assume that the prior probability of any given pronunciation $p(l(x) \mid x\ h)$ is known, for instance, by frequency counting, or just taking a uniform model over the elements of x. Our attention will now focus on the quantity $p(a(w) \mid l(x)\ h)$.

4.2. Acoustic events and their models

We will interpret an acoustic event $a(w)$ as a finite vector sequence $\langle \bar{a}_w{}^0 \ldots \bar{a}_w{}^{T-1} \rangle$, also written $\langle \bar{a}_w{}^i \rangle$, of d-dimensional feature vectors—that is, each $\bar{a}_w{}^i$ is an element of R^d. The list $\langle \bar{a}_w{}^i \rangle$ constitutes the *observation sequence*, the likelihood of which we desire.

In this paper, we will assume that the model $p(\cdot \mid l(x)\ h)$ is a continuous-density hidden Markov model (Jelinek, 1997, Chapter 2). Such a model consists of a set of states with identified initial and final states, a probability density function for each allowed state-to-state transition, and a matrix τ of transition probabilities. We establish notation as follows:

$Q = \{q_m\}$	a set of states
q_I, q_F	respectively initial, final states; drawn from Q
$\tau = \{\tau_{mn}\}$	the probability of transition $q_m \to q_n$
$\delta = \{\delta_{mn}\}$	a collection of densities, where δ_{mn} is the density associated with transition $q_m \to q_n$.

We refer to the collection $\langle Q,\ q_I,\ q_F,\ \tau,\ \delta\rangle$ as a *hidden Markov model* H. To distinguish between different models, say corresponding to lexemes $l(x)$ and $l(w)$, we will attach a subscript, and refer thus to H_x, its state set Q_x, a transition probability $\tau_{x\,mn}$, and so on. When a state has only two transitions, one of which is a self transition, we will write x_m in place of the self transition probability $\tau_{x\,mm}$ and $\bar{x}_m$ for the transition probability $\tau_{x\,mm'}$, $m \neq m'$.

The likelihood of a sequence of observations $p(\langle \bar{a}_w{}^i\rangle \mid l(x)\,h)$ is then taken as the sum over all paths from the initial to final state of the joint path and individual-observation probabilities. Since such a model serves to evaluate the likelihood of an observation sequence, we will refer to it as a *valuation model*.

This likelihood is exactly the number that we seek—and yet it is not. For we are proposing to found the training of a language model upon such numbers, and as we have already pointed out, the joint corpus $\langle \mathcal{C},\ \mathcal{A}\rangle$ at our disposal is a factor of 1000 smaller than a typical language model corpus. Moreover, the sequence $\langle \bar{a}_w{}^i\rangle$ constitutes one single pronunciation of the word w. This could very well be the sole instance, or one of the few instances, of the word in the corpus. It would be risky to rely upon so little data. What we would really like is a large number of such instances, ideally all pronounced in the same context h, which we may use collectively in computing $\arg\max_\theta P_\theta(\mathcal{C} \mid \mathcal{A})$.

For this reason, we adopt the strategy of synthesizing observation sequences corresponding to $a(w)$. To do so we use precisely the same model we would apply to evaluate the likelihood of an observation sequence, but operating with the model's densities and transition probabilities to generate data points. Though it has exactly the same form as an evaluation model, we will refer to such a model when used in this way as a *synthesizer model*. We are not the first to propose the use of synthetic data in speech recognition, and we note especially the contributions of McAllaster and Gillick (1999); McAllaster, Gillick, Scattone and Newman (1998).

4.3. Computing acoustic confusability

We now present our algorithm for computing acoustic confusability. Here is a sketch of the development. The algorithm uses the familiar hidden Markov model (HMM) formalism, and operates on pairs of lexemes. We proceed by constructing the product of each lexeme's associated HMM; the resulting *product machine* effectively represents all possible paths through either model. The arcs of this machine are labeled with a combination of transition probabilities and a synthetic measure of the confusability of densities, developed later. We then show how our proposed measure of acoustic confusability, $p(a(w) \mid l(x)\,h)$, comprising an exact sum over all possible state sequences, can be computed in a finite, closed-form expression. The resulting algorithm can be applied to hidden Markov models of arbitrary size and topology, subject only to practical limits of processor speed and memory size.

Our discussion proceeds in three stages. In Section 4.3.1 we consider the problem of synthesizing data according to one density and evaluating it according to another. This leads to a natural measure of confusability of densities, namely the cross entropy. Then in Section 4.3.2 we treat the case of hidden state, and give an efficient algorithm to compute the necessary sum over all path pairs.

4.3.1. Confusability of densities

Let us first consider a radically simplified version of the computation: suppose that for every acoustic event $a(w)$, the associated sequence $\langle \bar{a}_w{}^i \rangle$ has length 1, and that the dimension d of this single vector is also 1. In other words, $a(w)$ is identified with a single real number a_w. Likewise suppose that the valuation model $p(\cdot \mid l(x)\, h)$ has a single transition, with associated density $\delta_{l(x)h}$, hereafter abbreviated δ_x. Hence if $\mathcal{A}_w = \{a_{w1} \ldots a_{wN}\}$ is a corpus of one-dimensional, length-1 observations corresponding to N true pronounced instances of word w, then the likelihood of these observations according to the valuation model is

$$L(\mathcal{A}_w \mid \delta_x) = \delta_x(a_{w1}) \cdots \delta_x(a_{wN}). \tag{35}$$

Now we replace true observations with synthetic ones. Assume for a moment that word w has a single pronunciation $l(w)$, and consider a synthesized observation corpus $\hat{\mathcal{A}}_w = \{\hat{a}_{w1} \ldots \hat{a}_{wN}\}$, where the elements are iid random variables, distributed according to density $\delta_{l(w)h}(\cdot)$, hereafter abbreviated δ_w. Fix some finite interval $[-r, r]$, and imagine that it is divided into N subintervals $J_i = [\nu_i, \nu_i + \Delta\nu]$, where $\nu_i = -r + i\Delta\nu$ and $\Delta\nu = 2r/N$, where i runs from 0 to $N-1$. The expected number of elements of $\hat{\mathcal{A}}_w$ falling into J_i therefore goes as $\delta_w(\nu_i) \cdot \Delta\nu \cdot N$. We define the *synthetic likelihood* of this sequence as

$$L_{rN}(\hat{\mathcal{A}}_w \mid \delta_x) = \prod_{i=0}^{N-1} \delta_x(\nu_i)^{\delta_w(\nu_i)\cdot\Delta\nu\cdot N}. \tag{36}$$

Hence the per-event synthetic log likelihood is

$$S_{rN}(\hat{\mathcal{A}}_w \mid \delta_x) = \frac{1}{N} \log L_{rN}(\hat{\mathcal{A}}_w \mid \delta_x) = \sum_{i=0}^{N-1} \delta_w(\nu_i) \log \delta_x(\nu_i) \cdot \Delta\nu. \tag{37}$$

This is a Riemann–Stieltjes sum, as developed in Apostol (1957, Chapter 9). At this point we will assume that δ_w and δ_x are both mixtures of Gaussians. For mixtures of Gaussians the integral

$$\rho(\delta_w \mid \delta_x) = \int \delta_w \log \delta_x, \tag{38}$$

always exists and equals the limit of (37) as $\Delta\nu \to 0$ and $r \to \infty$. The quantity $\rho(\delta_w \mid \delta_x)$ is recognizable as the cross-entropy of δ_w and δ_x; it now has the additional interpretation as the *synthetic log likelihood of* δ_w *given* δ_x. That is, $\exp \rho(\delta_w \mid \delta_x)$ represents the per event likelihood, according to the model δ_x, of a corpus synthetically generated using the model δ_w. Based upon the reasoning that led to (38), we will treat this quantity as if it were a true likelihood. This substitution of synthetic for true likelihoods lies at the heart of our method.

When each Gaussian mixture δ_w and δ_x consists of a single Gaussian, that is $\delta_w(t) = \mathcal{N}(t; \mu_w, \Sigma_w)$ and $\delta_x(t) = \mathcal{N}(t; \mu_x, \Sigma_x)$, the quantity $\rho(\delta_w \mid \delta_x)$ can be computed exactly (Cover & Thomas, 1991, p. 30) and equals

$$\begin{aligned} \rho(\delta_w \mid \delta_x) = &-\tfrac{1}{2}\log(2\pi) - \tfrac{1}{2}\log\det\Sigma_w - \tfrac{1}{2}\mathrm{trace}(\Sigma_w^{-1}\Sigma_x) \\ &-\tfrac{1}{2}(\mu_x - \mu_w)^T \Sigma_w^{-1}(\mu_x - \mu_w). \end{aligned} \tag{39}$$

For general Gaussian mixtures no closed-form expression exists. However the synthetic likelihood, $\rho(\delta_w \mid \delta_x)$, can be numerically approximated using Monte Carlo methods (Press, Teukolsky, Vetterling & Flannery, 1999). Note that we can precompute ρ for all pairs of Gaussian mixtures appearing in our acoustic model. Because these quantities enter as constants in the acoustic confusability method developed later, the one-time cost of determining

TABLE I. Unsmoothed and smoothed confusabilities. Top: $l(x) =$ B AO S T AX N. Bottom: $l(x) =$ D AE L AX S

$l(x) =$ B AO S T AX N		$\log_{10} p_\lambda(l(w) \mid l(x))$	
w	$l(w)$	$\lambda = 0{\cdot}00$	$\lambda = -0{\cdot}86$
Boston	B AO S T AX N	−0·00	−0·57
Austin	AO S T AX N	−5·92	−1·40
Baden	B AO DX AX N	−10·59	−2·05
busted	B AH S T IX DD	−10·73	−2·07
bossed	B AO S TD	−11·54	−2·19

$l(x) =$ D AE L AX S		$\log_{10} p_\lambda(l(w) \mid l(x))$	
w	$l(w)$	$\lambda = 0{\cdot}00$	$\lambda = -0{\cdot}86$
Dallas	D AE L AX S	−0·00	−0·98
Dulles	D AH L AX S	−6·37	−1·87
Della	D EH L AX	−7·09	−1·97
ballots	B AE L AX TS	−8·85	−2·22
gala	G AE L AX	−8·91	−2·23

them does not impose a significant computational obstacle to computing acoustic confusability at the word level.

Since a probability density function can in general attain values larger than 1, complications can arise when mixing these quantities with the transition probabilities in the hidden Markov model framework. A solution is to introduce the *normalized synthetic likelihood of* δ_w *given* δ_x

$$\kappa(\delta_w \mid \delta_x) = \frac{\exp \rho(\delta_w \mid \delta_x)}{\sum_{w' \in V} \exp \rho(\delta_{w'} \mid \delta_x)}. \tag{40}$$

It is worth noting that this quantity is not symmetric in δ_w and δ_x. The synthetic log likelihood also has the property that any synthetic confusability measure founded upon $\rho(\delta_w \mid \delta_x)$, or a monotone function thereof, will decode faithfully. This is so since by the Gibbs inequality (Cover & Thomas, 1991, p. 29) we have $\rho(\delta_w \mid \delta_x) \leq \rho(\delta_w \mid \delta_w)$, with equality attained only when $\delta_w = \delta_x$.

The quantity $\exp \rho(\delta_w \mid \delta_x)$ represents the geometric mean of likelihoods δ_x for an infinite set of samples generated by δ_w, and we can write $\exp \rho(\delta_w \mid \delta_x) = \exp E_w[\log \delta_x]$, where E_w denotes the expected value with respect to δ_w. It might be argued that the arithmetic mean of likelihoods, $E_w[\delta_x]$, would be a more appropriate measure of confusability. However, some simple numerical experiments suggest this is not so. Specifically, we used the arithmetic mean measure to generate lists of confusable words. The results were not at all plausible, and frequently a word was not even listed as being acoustically similar to itself. (On this point, cf. Table I, computed with the expression we favor.) This cannot be so for a measure that decodes faithfully, a property that the arithmetic mean of likelihoods therefore does not have.

4.3.2. Confusability of hidden Markov models

We now develop a construction that defines a confusability measure between arbitrary hidden Markov models. This measure comprises observation sequences synthesized over all valid paths of all lengths, and yields an efficient algorithm that gives an exact result.

Our approach in many ways resembles the forward pass algorithm, used to determine a hidden Markov model's assignment of likelihood to a given sequence of observations, which

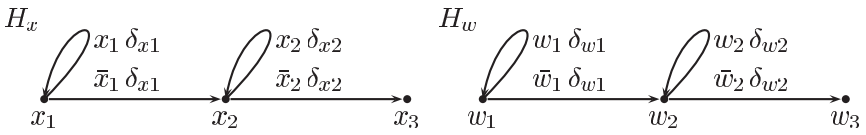

Figure 2. Models H_x and H_w. Here H_x has states $Q_x = \{x_1, x_2, x_3\}$, transition probabilities $x_1,\ \bar{x}_1,\ x_2$ and $\bar{x}_2$ associated with the arcs as shown, and densities δ_{x1}, δ_{x2}, δ_{x3}, associated as shown. The quantities for H_w are defined likewise.

we now briefly review. The forward pass algorithm operates on a trellis: a rectangular array of nodes with as many rows as the number of states in the model, and as many columns as the number of observations plus one. Each column of nodes constitutes one time slice of the trellis. Starting with likelihood 1 assigned to the initial state at the first time slice (and hence mass 0 assigned to all other states in this time slice), the algorithm assigns a likelihood to each state at each time, according to the equation

$$\alpha_s^{t+1} = \sum_{s'|s' \to s} \alpha_{s'}^t \cdot \tau_{s's} \cdot \delta_{s's}(o^t), \tag{41}$$

which we will call a *forward trellis equation*. Here α_s^t is the likelihood of state s at time t, $\tau_{s's}$ is a transition probability, and $\delta_{s's}(o^t)$ is the likelihood of the observation o^t recorded at time t. The notation $s' \mid s' \to s$ on the summation means that the sum is taken over all states s' with arcs incident on s. For a sequence of T observations, the value α_F^T computed for the final state F at the last time slice T is then the likelihood that the model assigns to the complete observation sequence.

As in the forward pass algorithm, to develop our confusability measure we will proceed by unrolling a state machine into a trellis, writing suitable forward trellis equations, and computing probabilities for trellis nodes. The key difference is that we do not operate with respect to a given sequence of true observations; here we are synthesizing the observations. This means that there is no natural stopping point for the trellis computation. What time T should we declare as the end of the synthesized observation sequence, and take the mass assigned to the final state in this time slice as the synthetic sequence likelihood? To resolve this problem we will operate on an *infinite trellis* and sum the probability assigned to the final state over *all* time slices.

We now explain the algorithm in detail. In what follows, H_x is the valuation model, and H_w is the synthesizer model. For purposes of illustration, we assume that both H_x and H_w are three-state models, with the topology, densities and transition probabilities as depicted in Figure 2. However the construction is entirely general, and there is no need to restrict the size or topology of either model.

From these two hidden Markov models, we define the product machine $H_{w|x}$ as follows. We begin by setting out some notation and definitions

$$\begin{aligned}
Q_{w|x} &= Q_w \times Q_x && \text{a set of states} \\
q_{w|x\,I} &= \langle q_{w\,I}, q_{x\,I} \rangle && \text{an initial state} \\
q_{w|x\,F} &= \langle q_{w\,F}, q_{x\,F} \rangle && \text{a final state} \\
\tau_{w|x\,\langle w_m, x_r \rangle \langle w_n, x_s \rangle} &= \tau_{w\,mn} \tau_{x\,rs} && \text{a set of transition probabilities.}
\end{aligned}$$

The states and transitions of this machine are depicted in Figure 3. Although superficially $H_{w|x}$ shares many of the characteristics of a hidden Markov model, it is not in fact a model of anything. In particular the arcs are not labeled with densities, from which observation likelihoods may be computed. Instead, we label an arc $\langle w_m, x_r \rangle \to \langle w_n, x_s \rangle$ with $\kappa(\delta_{w\,mn} \mid \delta_{x\,rs})$,

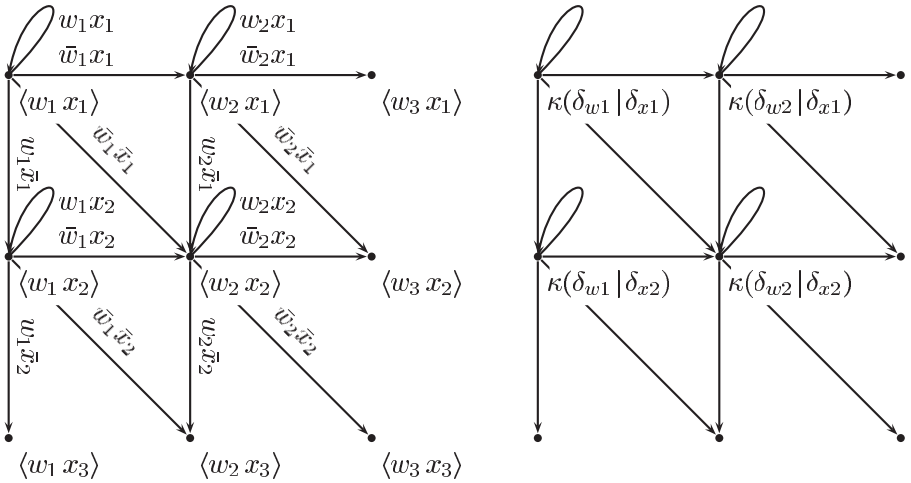

Figure 3. States, transitions and synthetic likelihoods of $H_{w|x}$. The graph of $H_{w|x}$ has nine states, comprising $Q_{w|x} = Q_w \times Q_x$. The arc $\langle w_m\, x_r\rangle \to \langle w_n\, x_s\rangle$ is drawn iff arcs $w_m \to w_n$ and $x_r \to x_s$ are drawn in H_w and H_x, respectively. The left panel shows the states and transition probabilities. The right panel shows the synthetic likelihoods. The value $\kappa(\delta_{wm} \mid \delta_{xr})$ is shared by all transitions that emanate from state $\langle w_m\, x_r\rangle$.

and treat this quantity as the likelihood, according to $\delta_{x\,rs}$, of observing a sample generated according to $\delta_{w\,mn}$.

Now observe that any path taken through the state diagram of $H_{w|x}$ is a sequence $\langle w^0\, x^0\rangle$, $\langle w^1\, x^1\rangle, \ldots$ of pairs of states of the original machines, H_w and H_x. There is a natural bijection between sequences $\pi_{w|x}$ of state pairs, and pairs of state sequences $\langle \pi_w\, \pi_x\rangle$. Moreover, every pair $\langle \pi_w\, \pi_x\rangle$, of valid paths of identical lengths in H_w and H_x, respectively, corresponds to a path in $H_{w|x}$, and conversely. Thus a computation that traverses all valid paths in $H_{w|x}$ comprises all pairs of same-length valid paths in the synthesizer and valuation models.

We proceed to construct a trellis for the state-transition graph of Figure 3, and to write appropriate forward trellis equations, with synthetic likelihoods in place of true observation probabilities. The left panel of Figure 4 shows two successive time slices in the trellis. The arcs drawn correspond to the allowed state transitions of $H_{w|x}$, as the reader is encouraged to confirm.

Now we derive the forward trellis equation for state $\langle w_1x_2\rangle$, as pictured in the right-hand panel of the same figure. Our aim is to obtain an expression for $\alpha^{t+1}_{\langle w_1x_2\rangle}$, the likelihood of arriving at this state at time $t+1$ by any path, having observed t frames of synthetic data for w, as evaluated by the densities of x. It is apparent from the diagram that there are only two ways that this can happen: via a transition from $\langle w_1x_1\rangle$, and via a transition from $\langle w_1x_2\rangle$ itself.

Let us suppose that the synthetic likelihood of arriving in state $\langle w_1x_1\rangle$ at time t by all paths is $\alpha^t_{\langle w_1x_1\rangle}$. The probability of traversing both transition $w_1 \to w_1$ in H_w and transition $x_1 \to x_2$ in H_x is $\tau_{w|x\,\langle w_1,\,x_1\rangle\langle w_1,\,x_2\rangle} = \tau_{w\,11}\tau_{x\,12} = w_1\bar{x}_1$, and the synthetic likelihood of the data corresponding to this transition pair is $\kappa(\delta_{w_1} \mid \delta_{x_1})$. Thus the contribution to $\alpha^{t+1}_{\langle w_1x_2\rangle}$ of all paths passing through $\langle w_1x_1\rangle$ at t is

$$\kappa(\delta_{w_1} \mid \delta_{x_1})\, w_1\bar{x}_1\, \alpha^t_{\langle w_1x_1\rangle}. \tag{42}$$

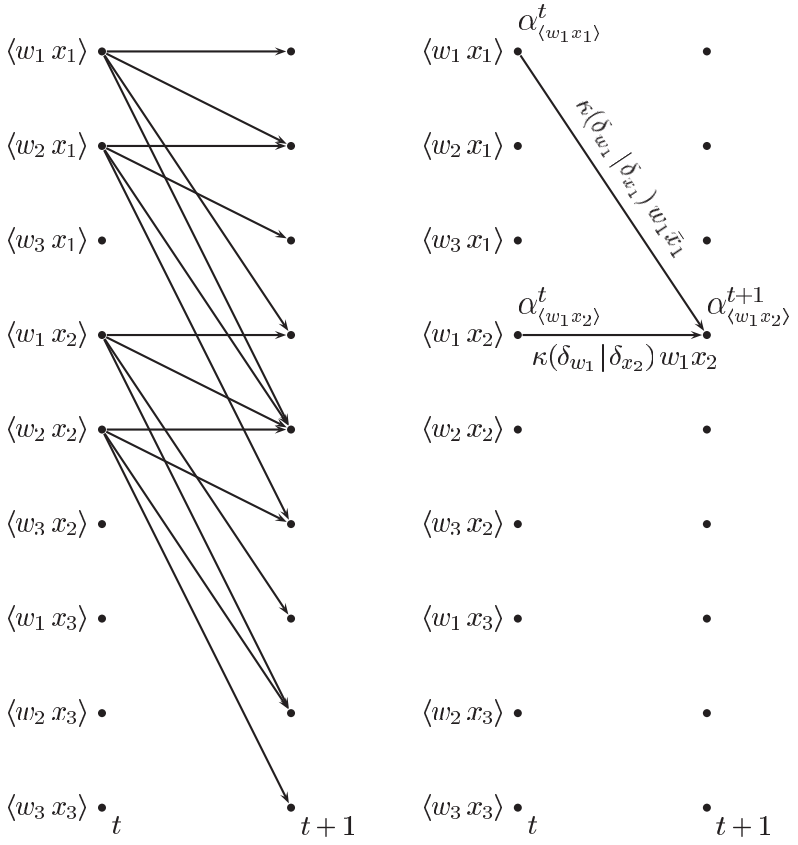

Figure 4. Trellis of $H_{w|x}$. The lefthand panel shows two successive slices, and legal transitions, of the trellis corresponding to $H_{w|x}$. Each bullet is a node of the trellis, and corresponds to the indicated state of $H_{w|x}$. For clarity of the diagram, we display names of the lefthand column of nodes only. The right-hand panel shows the derivation of the forward trellis equation for state $\langle w_1\, x_2\rangle$.

Likewise the contribution from paths passing through $\langle w_1 x_2\rangle$ at t is

$$\kappa(\delta_{w_1} \mid \delta_{x_2})\, w_1 x_2\, \alpha^t_{\langle w_1 x_2\rangle}. \tag{43}$$

Since these paths pass through different states at time t they are distinct, so their probabilities add and we have

$$\alpha^{t+1}_{\langle w_1 x_2\rangle} = \kappa(\delta_{w_1} \mid \delta_{x_1})\, w_1 \bar{x}_1\, \alpha^t_{\langle w_1 x_1\rangle} + \kappa(\delta_{w_1} \mid \delta_{x_2})\, w_1 x_2\, \alpha^t_{\langle w_1 x_2\rangle}, \tag{44}$$

the forward trellis equation for state $\langle w_1 x_2\rangle$. In a straightforward way, we can write such an equation for every state of $H_{w|x}$.

Now we make a crucial observation. Let us write $\bar{\alpha}^t$ for the distribution of probability mass across all nine states of $Q_{w|x}$ at time t, thus

$$\bar{\alpha}^t = \langle \alpha^t_{\langle w_1 x_1\rangle}\, \alpha^t_{\langle w_2 x_1\rangle}\, \alpha^t_{\langle w_3 x_1\rangle}\, \alpha^t_{\langle w_1 x_2\rangle}\, \alpha^t_{\langle w_2 x_2\rangle}\, \alpha^t_{\langle w_3 x_2\rangle}\, \alpha^t_{\langle w_1 x_3\rangle}\, \alpha^t_{\langle w_2 x_3\rangle}\, \alpha^t_{\langle w_3 x_3\rangle}\rangle^\top \tag{45}$$

and likewise $\bar{\alpha}^{t+1}$ for the same vector one timestep later. (The notation $\langle\cdots\rangle^\top$ means that each $\bar{\alpha}^t$ is a column vector.) The complete family of trellis equations can be expressed as

$$\bar{\alpha}^{t+1} = M\bar{\alpha}^t. \tag{46}$$

Here M is a square matrix of dimension $m \times m$, where $m = |Q_{w|x}| = |Q_w| \cdot |Q_x|$. We call M the *probability flow matrix*. Note that the elements of M do not depend at all upon t. The sparsity structure of M is a consequence of the allowed transitions of $H_{w|x}$, and the numerical values of its entries are determined by transition probabilities and synthetic likelihoods. For the example depicted in Figure 4, M is a 9×9 matrix given by

$$\begin{bmatrix}
\kappa_{11} w_1 x_1 & \kappa_{11} \bar{w}_1 x_1 & 0 & \kappa_{11} w_1 \bar{x}_1 & \kappa_{11} \bar{w}_1 \bar{x}_1 & 0 & 0 & 0 & 0 \\
0 & \kappa_{21} w_2 x_1 & \kappa_{21} \bar{w}_2 x_1 & 0 & \kappa_{21} w_2 \bar{x}_1 & \kappa_{21} \bar{w}_2 \bar{x}_1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \kappa_{12} w_1 x_2 & \kappa_{12} \bar{w}_1 x_2 & 0 & \kappa_{12} w_1 \bar{x}_2 & \kappa_{12} \bar{w}_1 \bar{x}_2 & 0 \\
0 & 0 & 0 & 0 & \kappa_{22} w_2 x_2 & \kappa_{22} \bar{w}_2 x_2 & 0 & \kappa_{22} w_2 \bar{x}_2 & \kappa_{22} \bar{w}_2 \bar{x}_2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

where we have used the shorthand notation $\kappa_{11} = \kappa(\delta_{w_1} \,|\, \delta_{x_1})$, $\kappa_{21} = \kappa(\delta_{w_2} \,|\, \delta_{x_1})$, $\kappa_{12} = \kappa(\delta_{w_1} \,|\, \delta_{x_2})$ and $\kappa_{22} = \kappa(\delta_{w_2} \,|\, \delta_{x_2})$. Note that M has only 16 non-zero elements, out of a total of 81. As we shall see in Section 4.5, this sparsity is a general property of the method, which can be exploited to obtain a fast algorithm.

Now we show how this formalism yields the desired result. By assumption, at time 0 all the probability mass in $\bar{\alpha}^0$ is concentrated on the initial state $\langle w_1 x_1 \rangle$, thus $\bar{\alpha}^0 = \langle 1 \ldots 0 \rangle^\top$. By iteration of Equation (46) we obtain the sequence of distributions

$$\bar{\alpha}^1 = M\bar{\alpha}^0, \qquad \bar{\alpha}^2 = M\bar{\alpha}^1 = M^2\bar{\alpha}^0, \qquad \bar{\alpha}^3 = M\bar{\alpha}^2 = M^3\bar{\alpha}^0, \ldots \tag{47}$$

or in general $\bar{\alpha}^t = M^t\bar{\alpha}^0$. Now let us ask, what is the total probability, over all time, of arriving in the final state $\langle w_3 x_3 \rangle$ of $H_{w|x}$? We will write $\xi_{w|x}$ for this quantity. By the set of equations in (47) we have

$$\xi_{w|x} = [(I + M + M^2 + \cdots)\bar{\alpha}^0]_{\langle w_3 x_3 \rangle} = [(I - M)^{-1}\bar{\alpha}^0]_{\langle w_3 x_3 \rangle}. \tag{48}$$

We have here assumed that the sum $(I + M + M^2 + \cdots)$ converges; in general this is not so. A sufficient condition for the sum to converge is that each eigenvalue λ of M satisfy $|\lambda| < 1$. [This follows from consideration of the repeated exponentiation of the Jordan canonical form of M; the interested reader may consult Herstein (1975, Section 6.6).]

In our experience we have not yet encountered a case where the sum diverges. This is not surprising: since all the entries of M are positive, and since by (40) the sum of the elements in any row of M is less than or equal to 1, each eigenvalue of M satisfies $|\lambda| \leq 1$ (Kreyszig, 1978, Problem 9, p. 469). If the slightly stronger criterion holds that each row sum is strictly less than 1, which in view of the construction of M is highly likely, then convergence is guaranteed. On the other hand if all the rows sum to 1 then by the Perron–Frobenius theorem $\lambda = 1$ is an eigenvalue and the sum does not converge. Finally, if the normalization step (40) is skipped, all bets are off and the sum will generally diverge.

Returning to the main line of development, observe that the vector $(I - M)^{-1}\bar{\alpha}^0$ is just the $\langle w_1 x_1 \rangle$ column of the matrix $(I - M)^{-1}$ and we seek the $\langle w_3 x_3 \rangle$ element of this vector. More generally, if $\bar{u}_I$ is $\bar{\alpha}^0$, which is to say an m-element vector with a 1 in the position corresponding to the initial state of $H_{w|x}$, and with 0s everywhere else, and if $\bar{u}_F$ is defined likewise, except with a 1 in the position corresponding to the final state of $H_{w|x}$, then

$$\xi_{w|x} = \bar{u}_F^\top (I - M)^{-1} \bar{u}_I. \tag{49}$$

We take this as the definition of the confusability of H_w given H_x. It is our algorithm's estimate of the likelihood, according to model H_x, of observing acoustics synthesized according to H_w.

We have treated H_w and H_x abstractly, but it should be clear that we intend for each one to represent a lexeme. Thus H_w is the hidden Markov model for some lexeme $l(w)$, and likewise H_x for $l(x)$. To exhibit this explicitly we will change notation slightly and write

$$\xi(l(w) \mid l(x)\, h) = \bar{u}_F^{\top}(I - M(l(w) \mid l(x)\, h))^{-1}\, \bar{u}_I. \tag{50}$$

The reader may be wondering why we introduced the new quantity ξ, rather than writing $p(l(w) \mid l(x)\, h)$ outright on the right-hand side of (50). The answer is that practical experience has shown us that (50) yields exceedingly small values. Much of the likelihood in acoustic space belongs to non-speech acoustic events, or so our models declare. Only a small amount is left to spread over the legitimate word sounds enumerated in the lexeme list B.

For this reason, using raw ξ values as probabilities in the computations detailed later in this paper rapidly exhausts the available precision of our computers. For this reason we renormalize the results of (50) via

$$p(l(w) \mid l(x)\, h) \stackrel{\text{def}}{=} \frac{\xi^{1+\lambda}(l(w) \mid l(x)\, h)}{\sum_{l(z)\in B} \xi^{1+\lambda}(l(z) \mid l(x)\, h)}. \tag{51}$$

The presence of the exponent λ is explained in Section 5.2.

4.4. Multiple pronunications and continuous speech

We have obtained a closed-form analytic expression for $p(l(w) \mid l(x)\, h)$; by Equation (34) we may combine results for the various $l(x) \in x$ to yield $p(l(w) \mid x\, h)$. However the word w itself may admit several pronunciations, though we assumed that there was only one, namely $l(w)$.

To discharge this assumption we will declare that $a(w)$ is a set comprised of all the $l(w) \in w$, and furthermore treat the various $l(w)$ as non-overlapping. It then follows that

$$p(a(w) \mid x\, h) = \sum_{l(w)\in w} p(l(w) \mid x\, h). \tag{52}$$

To extend these methods to the case of continuous speech, we operate with the quantity $p(a(h\ w\ r) \mid l(x)\, h)$. This is approximated by $p(a(h^{\epsilon}\ w\ r^{\epsilon}) \mid l(x)\, h)$, where the notations h^{ϵ} and r^{ϵ} mean that the full left and right contexts h and r are reduced to a few phones of context in either direction. We then operate as before, replacing $p(a(h^{\epsilon}\ w\ r^{\epsilon}) \mid l(x)\, h)$ by the synthetic quantity $p(l(h^{\epsilon}\ w\ r^{\epsilon}) \mid l(x)\, h)$, computed with the algorithm just developed.

4.5. Efficient computation of confusability

As previously discussed, the confusability is defined as $\xi_{w|x} = \bar{u}_F^{\top}(I - M)^{-1}\bar{u}_I$. Since $(I - M)$ is an $N \times N$ matrix, and this equation exhibits its inverse, it would seem that computing the confusability is an $O(N^3)$ operation. Recall N is the *product* of the number of states in H_w and the number of states in H_x, and typically lies between 200 and 300 for isolated words. Thus it would appear that determining the confusability of any two lexemes requires on the order of 10^7 arithmetic operations. If this were so, processing a complete vocabulary of lexeme pairs would be prohibitively expensive.

Fortunately, as we now demonstrate, it is possible to compute the desired confusability number in $O(N)$ operations; moreover additional simplifications can further reduce the workload. In the remainder of this section we explain these techniques. The development does not involve speech recognition per se, and readers not interested in computational details may skip to Section 5 with no loss of continuity.

In Section 4.5.1 we give (weak) conditions that must be satisfied to apply our algorithm. In Section 4.5.2 we describe the algorithm, and in Section 4.5.3 we show that it requires only $O(N)$ operations. Finally in Section 4.5.4 we discuss caching and thresholding, which further reduce the computation.

4.5.1. Conditions required to apply the algorithm

Two conditions must be satisfied to apply the algorithm. First, the synthesizer and evaluation hidden Markov models, used to construct the product machine, must have so-called "left-to-right" state graphs. The state graph of an HMM is left-to-right if it is acyclic except possibly for self-loops. The terminology "left-to-right" suggests that this idea has something to do with the way a state graph is drawn on a page, but in fact its meaning is the purely topological one just given. The HMMs that appear in speech recognition are almost always left-to-right, and indeed all the HMMs in this paper are left-to-right.

Second, the method described here is efficient in part because the maximum indegree (that is, the number of transitions or arcs impinging on any given node) of the synthesizer and evaluation models is a small number. The reason for this efficiency is explained further later. For the particular HMMs considered in Figure 2 the maximum indegree is 2, and in the discussion that follows, the technique will be explained as if this were so of every HMM. However, the method applies no matter what the true value of the maximum indegree is, though the efficiency of the method may be reduced somewhat.

Two important properties of the product machine follow from these conditions. First, because the state graphs of the synthesizer and valuation models are both left-to-right, so too is the state graph of the product machine that is formed from these two models. As a result, the states of the product machine may be assigned numbers, starting from 1 and proceeding sequentially through the number of states N of the machine, in such a way that whenever there is an arc from state number r to state number s, it follows that $r \leq s$. Such an assignment will be called a *topological numbering*. In particular, this numbering may be determined in such a way that 1 is the number of the initial state and N is the number of the final state.

Second, no state of the product machine has more than four arcs impinging on it, including self-loops. This is a consequence of the bounded indegree of the synthesizer and valuation models, whose product was taken to obtain the graph of the product machine. In general, if the synthesizer model has maximum indegree D_w and the valuation model has maximum indegree D_x, then the maximum indegree of any state of the product machine is $D_w \times D_x$. For instance, in the examples of Figure 2, $D_w = D_x = 2$, and hence the product machine of Figure 3 has maximum indegree $D_w \times D_x = 4$.

The significance of this bound is as follows. It is evident from Figure 3 that not every possible state-to-state transition in the product machine is present. This means that only certain elements of the probability flow matrix may be non-zero. Indeed, the maximum number of non-zero entries in any row of M is the maximum indegree of the product machine. Thus, carrying this example a little further, the maximum number of non-zero entries in any row of M is four. As a result the total number of non-zero entries in the entire matrix M is no

greater than $D_w \times D_x \times N$, where N is the total number of states in the product machine. This property will be made use of later.

4.5.2. Description of the algorithm

Return now to Equation (49). This expression selects a single element of matrix $(I - M)^{-1}$, namely the one that lies in the matrix column that corresponds to the initial state of the product machine, and in the row that corresponds to the final state of the machine. For the product machine example considered earlier, these are, respectively, the first column ($j = 1$) and the last row ($i = 9$).

Note that in computing the quantity $\xi_{w|x}$, the κ quantities for all pairs of densities may be computed beforehand. Thus only the computation of the desired element of $(I - M)^{-1}$ is left. To compute this element, recall that the inverse of a matrix may be determined by a sequence of elementary row or column operations (see, for instance, Anton, 1973, Section 1.7). The following explanation assumes that row operations are performed; the explanation could easily be modified so that column operations are performed.

As explained in Anton (1973), to perform a matrix inversion, it suffices to start with a subject matrix, in this case $(I - M)$, and perform a series of elementary row operations converting the subject matrix to I. When this same series of operations is applied in the same order to an identity matrix I, the result is the inverse of the original subject matrix. The difficulty with this method is that for a general $N \times N$ matrix, it requires $O(N^3)$ arithmetic operations.

We now explain how, by exploiting both the sparsity structure of $(I - M)$ and the need for just a single element of its inverse, we can obtain a very substantial reduction in computation, compared to the general method just cited. As an instructive exercise, consider the inversion of a 4×4 matrix $(I - M)$, corresponding to some product machine. By the discussion in Section 4.5.1, it is assumed that the nodes in the product machine can be topologically numbered, so that the matrix M, and hence also $(I - M)$, is lower diagonal, that is, all non-zero elements lie on or below the main diagonal. Assume that such an ordering has been performed and denote the elements of $(I - M)$ by ϕ_{ij}, where the non-zero elements satisfy $1 \leq j \leq i \leq N$. Note that each $\phi_{ii} > 0$, since otherwise we would have $m_{ii} = 1$, which would entail probability-one self-loops within both H_w and H_x. Assume also, with no loss of generality, that the index 1 corresponds to the start state of the product machine, and the index N corresponds to the final state of the product machine. This entails that the desired element of $(I - M)^{-1}$ for these purposes is the row N, column 1 entry.

Now we explain how to apply a modification of the method of elementary row operations, and show how to obtain the desired simplification in computation. First, write down an augmented matrix $[(I - M) \mid I]$, consisting of $(I - M)$ and I written side-by-side as shown below.

$$[(I - M) \mid I] = \left[\begin{array}{cccc|cccc} \phi_{11} & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \phi_{21} & \phi_{22} & 0 & 0 & 0 & 1 & 0 & 0 \\ \phi_{31} & \phi_{32} & \phi_{33} & 0 & 0 & 0 & 1 & 0 \\ \phi_{41} & \phi_{42} & \phi_{43} & \phi_{44} & 0 & 0 & 0 & 1 \end{array}\right]. \tag{53}$$

Next convert the element ϕ_{11} to unity, by multiplication of the first row by $d_1 := 1/\phi_{11}$; recall that this is an elementary row operation. By performing similar operations on each row

of the matrix—that is, by multiplying row i by $d_i := 1/\phi_{ii}$—we obtain

$$[(I - M) \mid I] \sim \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & r_1 := d_1 & \cdot & \cdot & \cdot \\ b_{21} & 1 & 0 & 0 & 0 & \cdot & \cdot & \cdot \\ b_{31} & b_{32} & 1 & 0 & 0 & \cdot & \cdot & \cdot \\ b_{41} & b_{42} & b_{43} & 1 & 0 & \cdot & \cdot & \cdot \end{array}\right]. \tag{54}$$

Formally, $b_{ij} = \phi_{ij}/\phi_{ii}$ for $1 \le j \le i \le N$. Here the symbol $\sim$ is used to denote similarity by means of a series of elementary row operations. The quantity r_1 is defined as shown.

Note that a dot has been placed in certain positions of the augmented matrix, replacing the 0s or 1s previously exhibited there. This is to indicate that operations upon these positions need not be performed, as these positions have no effect upon the outcome of the computation. This is because, by the earlier discussion, only the row 4, column 1 entry of $(I - M)^{-1}$ is required, and this quantity depends only upon operations performed in the first column of the right submatrix of $[(I - M)|I]$. This observation eliminates the need to perform any elementary row operations on the remaining $N-1$ columns of the right submatrix. Since this eliminates $O(N^3)$ arithmetic operations, compared to the general method for matrix inversion, it is a very significant simplification. [Of course, removing $O(N^3)$ operations from an algorithm may still yield an $O(N^3)$ algorithm, but as we show later this is not the case.]

Returning to the development of the algorithm, next we perform elementary row operations to zero out the off-diagonal elements of $(I - M)$ one column at a time, starting with the leftmost column and proceeding through the right-most. For example, operating now upon the leftmost column, to clear the b_{21} element, multiply the first row by $-b_{21}$, and add it to the second row. Likewise to clear the b_{31} element, multiply the first row by $-b_{31}$ and add it to the third row, and so on through each succeeding row. After completing all operations necessary to clear the first column, we obtain

$$[(I - M) \mid I] \sim \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & r_1 := d_1 & \cdot & \cdot & \cdot \\ 0 & 1 & 0 & 0 & r_2 := -b_{21}r_1 & \cdot & \cdot & \cdot \\ 0 & b_{32} & 1 & 0 & -b_{31}r_1 & \cdot & \cdot & \cdot \\ 0 & b_{42} & b_{43} & 1 & -b_{41}r_1 & \cdot & \cdot & \cdot \end{array}\right]. \tag{55}$$

Having completed the operations to clear the first column, define the quantity r_2 as shown.

Similar sequences of operations are performed to clear the second and third columns. The method ends with

$$[(I - M) \mid I] \sim \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & r_1 := d_1 & \cdot & \cdot & \cdot \\ 0 & 1 & 0 & 0 & r_2 := -b_{21}r_1 & \cdot & \cdot & \cdot \\ 0 & 0 & 1 & 0 & r_3 := -b_{32}r_2 - b_{31}r_1 & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 1 & r_4 := -b_{43}r_3 - b_{42}r_2 - b_{41}r_1 & \cdot & \cdot & \cdot \end{array}\right]. \tag{56}$$

Note that the original subject matrix $(I - M)$, on the left-hand side of the augmented matrix, has been reduced to the identity, and hence its inverse (or rather the first column of the inverse, since operations on the other columns were intentionally not performed) has been developed in the right half of the augmented matrix. Thus r_4 is the desired element of $(I - M)^{-1}$. Moreover, by comparison of the expressions for r_1–r_4 with the original matrix elements, it can be seen that

$$r_1 = \frac{1}{\phi_{11}} \tag{57}$$

$$r_2 = \frac{-\phi_{21}r_1}{\phi_{22}} \tag{58}$$

$$r_3 = \frac{-\phi_{32} r_2 - \phi_{31} r_1}{\phi_{33}} \tag{59}$$

$$r_4 = \frac{-\phi_{43} r_3 - \phi_{42} r_2 - \phi_{41} r_1}{\phi_{44}}. \tag{60}$$

It is apparent from this series of expressions that the value r_4 depends only on computations performed within the first column of the right submatrix, utilizing either elements of the original subject matrix $(I - M)$, or the values r_1–r_4. Thus there is no need to operate upon the $(I - M)$ half of the augmented matrix at all. The only quantities of importance are the ones that are used in the determination of r_4, and these are precisely the expressions for r_1–r_3, or the original non-zero elements of $(I - M)$. Thus while inspection of the left half of the augmented matrix will determine *which* operations will be performed on the first column of the right half, no actual arithmetic will be performed on the left half. This yields another saving of $O(N^3)$ operations, compared to a general matrix inversion.

Consider now the effect upon the steps of the preceding description if some matrix entry ϕ_{ij} (and hence also b_{ij}, after division by ϕ_{ii}) in a particular row i were already zero, due to the sparsity structure of $(I - M)$. It is clear that, in such a case, one need not perform any elementary row operation at all. (For after all, the only reason for operating upon row i in that column is to reduce b_{ij} to 0, and here one is in the salubrious position of finding this element already zeroed.) Since the number of non-zero elements in any given column of $(I - M)$ is bounded above by four (in the general case by $D_w \times D_x$), this means that no more than four (in general, $D_w \times D_x$) elementary row operations will be generated by inspection of any given column. This is another significant saving, since it reduces $O(N)$ arithmetic operations per column to a constant $O(1)$ operation, depending only upon the topologies of the synthesizer and valuation machines.

The algorithm can now be stated in complete detail. Let ϕ_{ij} denote the row i column j element of the $N \times N$ matrix $(I - M)$. The recurrence formulae are:

$$r_1 = \frac{1}{\phi_{11}} \tag{61}$$

$$r_i = \frac{\left\{ -\sum_{k=1}^{i-1} \phi_{ik} r_k \right\}}{\phi_{ii}} \qquad \text{for} \quad i = 2, \ldots, N \tag{62}$$

where the curly brace notation means that the sum extends only over indices k such that $\phi_{ik} \neq 0$. The element sought is r_N and so by the equations displayed above it suffices to compute r_i for $i = 1, \ldots, N$.

4.5.3. Asymptotic analysis

Let us return to the general recurrence formulae presented. We determine the number of arithmetic operations (additions, multiplications or divisions) entailed by these formulae. Consider the expression for the numerator of r_i, for $i = 2, \ldots, N$, which is $\{-\sum_{k=1}^{i-1} \phi_{ik} r_k\}$. Recall that the curly braces mean that the sum proceeds over non-zero entries ϕ_{ik} of the matrix $I - M$.

As previously demonstrated, the total number of non-zero entries in M is bounded above by $D_w D_x N$, where N is the number of states of the product machine, and also the number of rows (and columns) of M. Thus the total number of multiplications and additions required to compute all numerators of all the r_i values, for all $i = 1, \ldots, N$, is no more than $D_w D_x N$. To this we also add a total of N divisions since, to obtain each final r_i value, the numerator

must be divided by ϕ_{ii} in each case. Hence the total number of arithmetic operations of all kinds (that is, addition, multiplication or division) required to compute the desired value is no more than $N(1 + D_w D_x)$, which is an $O(D_w D_x N)$ expression. This compares favorably to $O(N^3)$, the complexity of a generic matrix inversion algorithm.

To appreciate the impact of this asymptotic improvement, recall that N is determined as the product of the number of states in the synthesizer model and the number of states in the valuation model. Since a typical word will contain five phonemes, and each phoneme is modeled by a three-state HMM, this means that N typically attains values of about $(5 \times 3)^2 = 225$. Thus we are reducing a computation that requires more than $225^3 = 11\,390\,625$ arithmetic operations to $(4 + 1) \times 225 = 1125$ arithmetic operations.

4.5.4. Caching and thresholding

The final refinements to our algorithm consist of caching and thresholding. Caching is a method for partially reusing previously computed results. Thresholding is a method for early termination of the computation.

First we describe caching. If one wishes to compute $\xi_{w|x}$ for all $w \in V$, which will typically be the case, there is a saving in reusing computations for similar words. Consider two words w_1 and w_2, and suppose their pronunciations are identical through some initial prefix of phonemes. If the synthesizer machines H_{w_1} and H_{w_2} are identical for states $i = 1, \ldots, m$, then for any fixed valuation model x the entries ϕ_{ij} of appearing in $(I - M)$ for rows 1 to m, and all columns, will be identical. Hence the values $r_1, \ldots, r_m$ will be identical, and once they have been computed they can be stored and reused. This gives an improvement for long words that have pronunciations that begin in the same way, such as *compute* and *computability*. In this case, the full computation for *compute* can be reused for *computability* (assuming that right context is being ignored in determining HMM densities).

Now we describe thresholding. The idea of thresholding is that if one is not interested in computing $\xi_{w|x}$ for lexemes w and x that are highly distinguishable, then it is not necessary to complete the computation of $\xi_{w|x}$ when it is certain that it will be sufficiently small. In general $\xi_{w|x}$ is thrown out if it is less than $\epsilon\xi_{x|x}$ for some user specified ϵ. The implicit assumption here is that $\xi_{x|x}$ is likely to be large, compared to some arbitrary word w that is acoustically dissimilar to x. For this to be of any use, there needs to be a way of rapidly estimating an upper bound for $\xi_{w|x}$ and stopping the computation if this upper bound lies below $\epsilon\xi_{x|x}$.

To do this, first note that the value of any given r_i in the recurrence equations (61) and (62) may be written as a sum of products between fractions ϕ_{ik}/ϕ_{ii} and previously computed r_i values. Thus we have the bound

$$|r_i| \leq (i - 1)\left(\max_{1 \leq k \leq i} \frac{|\phi_{ik}|}{|\phi_{ii}|}\right)\left(\max_{\{k<i\}} r_k\right), \tag{63}$$

where the curly braces indicate that the second max need run only over those k values for which $\phi_{ik} \neq 0$. By using this bound, as the computation of $\xi_{w|x}$ proceeds, it is often possible to determine at some intermediate point that $\xi_{w|x}$ will never attain a value greater than the threshold $\epsilon\xi_{x|x}$. At this point the computation of $\xi_{w|x}$ is abandoned, yielding another substantial computational saving. The confusability $\xi_{w|x}$ is then assigned some small nominal predetermined value.

5. Performance statistics

In this section we investigate the use of confusability in predicting the performance of language models. We begin by introducing a new statistic, the synthetic acoustic word error rate (SAWER). Then we discuss the adjustment of the exponent λ, which appears in the acoustic encoding probabilities that underlie acoustic perplexity. Finally we present experimental evidence comparing lexical perplexity, acoustic perplexity and SAWER as measures of language model performance. In our experiments, SAWER was the most reliable of the three statistics.

5.1. Synthetic acoustic word error rate

As argued earlier, the acoustic encoding probability $p(a(w) \mid x\, h)$ is the probability, according to the acoustic models for word x in context h, of observing acoustic data $a(w)$. Thus it is natural to speak of $p_\theta(w \mid a(w)\, h)$, obtained by Bayes' theorem via (20), as the *acoustic decoding probability*.

In fact this number does not represent the probabilistic operation of any real decoder that we know of. It is the probability, according to the models appearing in (20), that word w was spoken, given lexical history h and acoustics $a(w)$. However, let us for the moment treat it as an estimate of the probability of decoding word w, and ask what conclusions we can draw.

We proceed to define a random variable X_i, associated with position i of the joint corpus $\langle \mathcal{C},\ \mathcal{A} \rangle$. Suppose for the moment that $\mathcal{A}$ contains true and not synthesized acoustics, and that we have decoded the entire corpus. Let the sequence $\langle X_i \rangle$ represent the outcome of this decoding experiment, as follows: X_i is 0 if acoustics $a(w_i)$ are decoded correctly as w_i, and X_i is 1 otherwise. Let $N = |\mathcal{C}|$ be the size of the corpus. Then for an assignment of 0s and 1s to $\langle X_i \rangle$, corresponding to some actual decoding of acoustics $\mathcal{A}$, the quantity $\sum_{i \in \mathcal{C}} X_i / N$ is the *true word error rate*, ignoring insertions and deletions.

We now consider this statistic for the case of synthesized acoustics, with the behavior of the decoder modeled by the quantity $p_\theta(w_i \mid a(w_i)\, h_i)$. If this is nominally the probability of correctly decoding w_i from acoustics $a(w_i)$, then its complement $1 - p_\theta(w_i \mid a(w_i)\, h_i)$ is the probability of decoding $a(w_i)$ incorrectly. Thus the *expected word error rate* according to this model is

$$S_A(P_\theta, \mathcal{C}, \mathcal{A}) = E_{p_\theta}\left[\sum_{i \in \mathcal{C}} X_i / N\right] = \sum_{i \in \mathcal{C}} E_{p_\theta}[X_i]/N \tag{64}$$

$$= \sum_{i \in \mathcal{C}} (1 - p_\theta(w_i \mid a(w_i)\, h))/N, \tag{65}$$

where we have used $E_{p_\theta}[X_i] = 0 \cdot p_\theta(w_i \mid a(w_i)\, h) + 1 \cdot (1 - p_\theta(w_i \mid a(w_i)\, h))$. We refer to S_A as the *synthetic acoustic word error rate*, or SAWER.

5.2. Adjustment of smoothing parameter λ

In Equation (51) we introduced the parameter λ on grounds that the raw confusabilities ξ are too sharp. At the time we presented no justification for this claim. We do so now, and also explain how we propose to determine λ. Formula (51), used to adjust the confusability scores, closely resembles the posterior probability calculation recently used for confidence scoring (Wessel *et al.*, 1998; Mangu, Brill & Stolcke, 1999; Evermann & Woodland, 2000). Indeed the need to scale acoustic loglikelihoods for combination with language model log probabilities has long been known, and this was what guided us to consider Equation (51).

For justification, we need look no further than the raw confusabilities of words that are frequently decoded wrongly. Consider, for instance, the words *Boston* and *Dallas*. Table I shows the five most confusable lexemes of each word, computed from (51) both without ($\lambda = 0{\cdot}0$) and with ($\lambda = -0{\cdot}86$) smoothing.

Inspecting the unsmoothed logprobs ($\lambda = 0{\cdot}0$) reported in the table, we see a gap between $l(w) =$ B AO S T AX N and the next most confusable lexeme $l(w) =$ AO S T AX N of almost six orders of magnitude. In other words, the estimated probability of decoding *Austin* when *Boston* was said is about 1 000 000 times smaller than decoding the word correctly. But in fact, *Austin* is a frequent misdecoding of *Boston*.

This is evident in the value of $p_{\lambda=0}(\textit{Boston} \mid \textit{Boston})$, which is so close to unity that its logarithm, though negative, is zero to a precision of better than 0·01; its true logarithm is in the vicinity of $-0{\cdot}000001$. Thus almost no mass is accorded to any other lexeme in the distribution, even taken all together. A similar observation holds for the distribution $p_{\lambda=0}(l(w) \mid \textit{Dallas})$.

These gaps do not accord with experience, and so we regard the raw model as excessively sharp. We believe this excessive sharpness arises because of a well-known weakness of hidden Markov models, which is that they treat successive acoustic observations as independent events. Since these observations are of course well-correlated, this results in a severe underestimate of the likelihood of true observation sequences. Moreover, since our synthesis scheme generates observations independently as well, our method exhibits this weakness in spades.

The solution we adopted was to introduce the smoothing parameter λ in (51). We then decoded an independent corpus $\mathcal{H}$ to obtain a true word error rate, computed the SAWER $S_A(P_{\theta\lambda}, \mathcal{H}, \mathcal{A})$ on the same corpus, and adjusted λ to match S_A to the true word error rate. This yielded $\lambda = -0{\cdot}86$, for an exponent of $1 + \lambda = 0{\cdot}14$. Some of the resulting smoothed lexeme confusabilities, which are much more plausible, are exhibited in the right-hand column of Table I. We use this value of λ for the experiments reported later.

5.3. Predictive power

We now exhibit results for three empirical measures of language model performance: lexical perplexity Y_L, acoustic perplexity Y_A and synthetic acoustic word error rate S_A. We tested each measure on three independent test corpora, respectively SOB (11 180 total words, 10 speakers, office dictation), NRR (9060 total words, five speakers, IBM ViaVoice product consumer data) and SPT (7429 words, 10 speakers, spontaneous speech). For each test corpus, we evaluated the same five language models. An evaluation consisted of determining the true word error rate, by decoding with each language model, and also computing values of the three measures, via (1), (14) or (65), using the true text and synthesized acoustics for each corpus.

The models used in these measurements were all linearly interpolated trigram language models (Jelinek, 1997), computed over the same fixed 64K word vocabulary. They were constructed from three corpora, of respective sizes 0·85 billion words (GW) of text, 1·0 GW of text, and 1·6 GW of text. The first two corpora contained a wide variety of text, including newswire, broadcast transcripts, patent text, office correspondence, and other sources; the second corpus was an expanded version of the first. The third corpus was comprised of 3 years of text drawn from information industry periodicals, and unrelated to the first two corpora. The language models differed as well in the numbers and characteristics of trigrams and bigrams that were retained in building the individual models.

TABLE II. Sample correlation coefficient r for test data sets. The values displayed are the correlation coefficients between the given statistic (respectively lexical perplexity, acoustic perplexity, and synthetic acoustic word error rate) and the true word error rate for the given corpus

Corpus	$Y_L(P_\theta, \mathcal{T}, \mathcal{A})$	$Y_A(P_\theta, \mathcal{T}, \mathcal{A})$	$S_A(P_\theta, \mathcal{T}, \mathcal{A})$
SOB	0·918	0·979	0·989
NRR	−0·562	−0·402	0·934
SPT	−0·995	−0·921	0·951

Figure 5 displays the results of these tests. Each vertical axis gives the true word error rate, and each horizontal axis gives one of the three statistics listed earlier. A statistic that correlates well with word error rate will have a graph that slopes more or less directly from lower left to upper right. It is clear from appearances alone that the SAWER statistic is a much better predictor of model performance than either of the other two. Moreover Table II, listing the sample correlation coefficient (Hoel, 1984, Section 7.1) for each statistic against word error rate, for each data set, provides empirical confirmation of this claim. However, it should be noted that our viewpoint is not uniformly held. In particular see Klakow and Peters (0000) for a defense of lexical perplexity as a training criterion.

It is not so surprising to see that SAWER correlates well with word error rate, since SAWER is constructed to approximate the word error rate (ignoring insertion and deletion errors, which may be treated in part by including a silence lexeme in the confusability computation). The SAWER statistic is more costly to compute than lexical perplexity, but is still quite a bit faster than decoding the entire test data. Moreover it can be used as an objective function when training a language model.

6. Training of language models

We have shown that acoustic perplexity and SAWER—especially the latter—are better predictors of language model performance than lexical perplexity. This has led us to investigate the adoption of one or the other as the objective function to be minimized in the training of language models.

In this section we provide the mathematical and computational tools for pursuing this program. Unfortunately there is no direct analytic expression for the global minimum of either of our measures. In Section 6.1 we prove a theorem that yields an iterative algorithm for finding the minimum. However as formulated this algorithm is impractical, so in Section 6.2 we describe a numerical technique, founded upon the idea of the theorem, that is practical. Finally in Section 6.3 we describe a precomputation scheme that yields an additional speedup. The result is an algorithm that can be used to train a unigram language model.

6.1. Steepest descent method

In this section we give an iterative algorithm for training a language model by optimization of the synthetic acoustic word error rate. Though we will not develop the relationship in detail, in fact the same algorithm, with appropriate changes, can be used to optimize the acoustic perplexity.

For clarity we treat the case of discrete speech; however, our methods apply to continuous speech as well. Our starting point is Equation (65). We seek the language model family

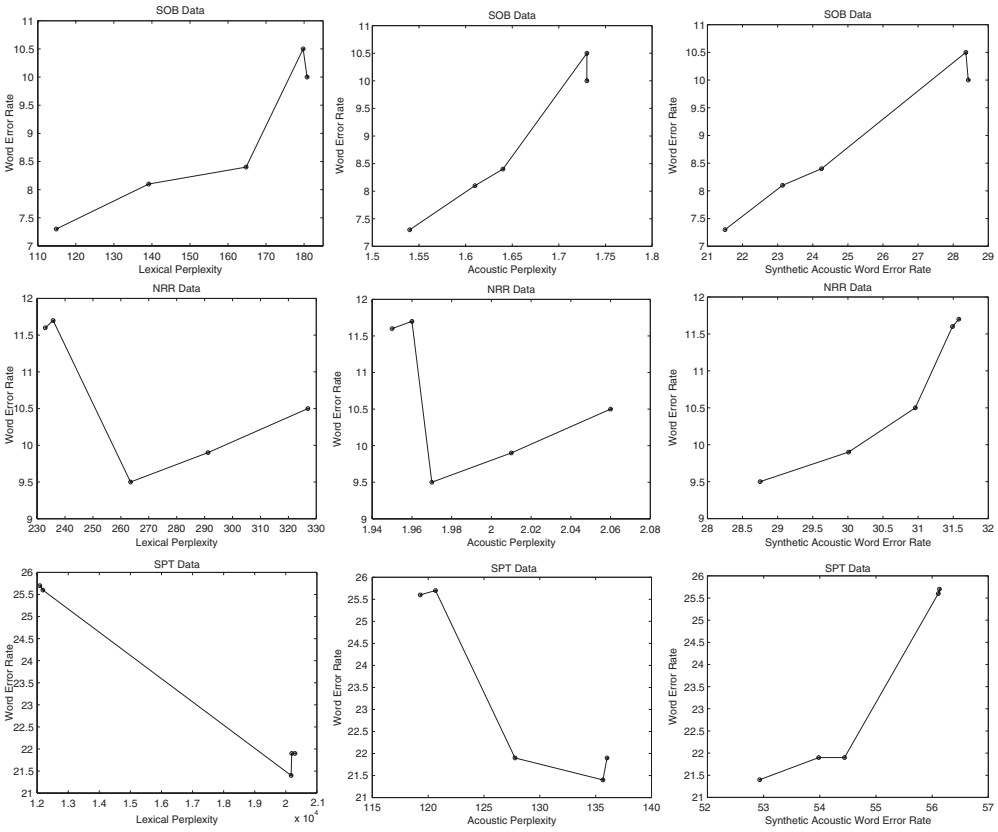

Figure 5. Comparison of lexical perplexity, acoustic perplexity and synthetic acoustic word error rate. The lefthand column of graphs shows the relation between lexical perplexity and true word error rate, for each of five different language models, as computed on three test corpora. The middle column shows this relation for acoustic perplexity, and the right-hand column for synthetic acoustic word error rate, both against true word error rate, for the same five language models, and the same three test corpora. Top row: SOB. Middle row: NRR. Bottom row: SPT.

$P_\theta = \{p_\theta(w \mid h)\}$, that minimizes $S_A(P_\theta, \mathcal{C}, \mathcal{A})$, where the parameters θ are raw language model probabilities $p_\theta(w|h)$. We hereafter write θ_{wh} for $p_\theta(w|h)$.

Our first observation is that minimizing $S_A(P_\theta, \mathcal{C}, \mathcal{A})$ is equivalent to maximizing $1 - S_A(P_\theta, \mathcal{C}, \mathcal{A})$. Moreover by collecting terms with identical history h, we obtain

$$1 - S_A(P_\theta, \mathcal{C}, \mathcal{A}) = \frac{1}{|\mathcal{C}|} \sum_{h \in \mathcal{C}} \left(\sum_{w \in V} c(w, h) \frac{p(a(w)|w\ h)\theta_{wh}}{\sum_{x \in V} p(a(w)|x\ h)\theta_{xh}} \right), \tag{66}$$

where $c(w, h)$ is the number of times w appears in context h, and the outer sum runs over distinct *histories* in $\mathcal{C}$. Because the probability distributions $p_\theta(w \mid h) = \theta_{wh}$ are independent for distinct h, it suffices to maximize the parenthesized sum in (66) separately for each value of h. We proceed to do this now.

Fix the history h to some definite value, and define the function

$$f(\theta) = \sum_{w \in V} c_w \frac{a_w \theta_w}{\sum_{x \in V} b_{wx} \theta_x}. \tag{67}$$

Here $c_w = c(w, h)$, $a_w = p(a(w) \mid w\, h)$ and $b_{wx} = p(a(w) \mid x\, h)$; moreover θ is just the vector $\langle \theta_w \rangle = \langle \theta_{wh} \rangle$ for fixed h, as w runs through V. Comparison shows that $f(\theta)$ is the parenthesized sum in (66) for the given h. Note that $f(\theta)$ satisfies $f(t\theta) = f(\theta)$ for all $t > 0$. Writing $m = |V|$, we proceed to establish the following lemma.

Lemma 6.1: *Let $f : \mathbf{R}^m \to \mathbf{R}$ be a $C^1(\mathbf{R}^m)$ function satisfying $f(t\theta) = f(\theta)$ for all $t > 0$. Then*

$$\sum_{i \in V} \theta_i \frac{\partial f}{\partial \theta_i}(\theta) = 0 \quad \forall \theta \in \mathbf{R}^m. \tag{68}$$

Proof: Define the function $g(t) = f(t\theta)$. Since $g(t)$ is a constant function we have $0 = \frac{dg}{dt}|_{t=1} = \frac{d}{dt}\{f(t\theta)\}|_{t=1} = \sum_{i \in V} \theta_i \frac{\partial f}{\partial \theta_i}(\theta)$. □

A geometric proof follows from the observation that f being constant along the direction θ implies that the gradient $\nabla f(\theta)$ must be orthogonal to θ.

The following theorem gives us a method to locate incrementally larger values for $f(\theta)$. It specifies a direction in which we are guaranteed to find a better value, unless we are at a boundary point, or a point where $\nabla f = 0$.

Theorem 6.1: *Let $f : \mathbf{R}^m \to \mathbf{R}$ be a $C^2(\mathbf{R}^m)$ function such that $f(t\theta) = f(\theta)$ for all $t > 0$. Consider θ satisfying $\sum_{i \in V} \theta_i = 1$ and $\theta_i \geq 0\ \forall i$. Suppose as well that for some $i \in V$, both $\frac{\partial f}{\partial \theta_i}(\theta) \neq 0$ and $0 < \theta_i < 1$ hold. Define $\hat{\theta}_i^\epsilon = \theta_i + \epsilon \theta_i \frac{\partial f}{\partial \theta_i}(\theta)$. Then there exists $\epsilon > 0$ such that the following three properties hold*

$$\textstyle\sum_{i \in V} \hat{\theta}_i^\epsilon = 1, \tag{69}$$

$$\hat{\theta}_i^\epsilon \geq 0 \quad \forall i \in V \tag{70}$$

and

$$f(\hat{\theta}^\epsilon) > f(\theta). \tag{71}$$

Proof: The proof of (69) follows from Lemma 6.1: we have $\sum_{i \in V} \hat{\theta}_i^\epsilon = \sum_{i \in V} \theta_i + \epsilon \sum_{i \in V} \theta_i \frac{\partial f}{\partial \theta_i}(\theta) = 1 + 0 = 1$.

For (70), observe that since $\hat{\theta}_i^\epsilon = \theta_i(1 + \epsilon \frac{\partial f}{\partial \theta_i})$ and $\theta_i \geq 0$, it suffices that the parenthesized quantity be non-negative. If $\partial f / \partial \theta_i \geq 0$ this is immediate. If $\partial f / \partial \theta_i < 0$, it suffices that $\epsilon < -1/(\partial f / \partial \theta_i)$. Hence by choosing ϵ sufficiently small all the inequalities of (70) will be satisfied.

Finally to establish (71), by Taylor's theorems (Apostol, 1957, Theorems 6–22)

$$f(\hat{\theta}^\epsilon) = f(\theta) + \sum_{i \in V} (\hat{\theta}_i^\epsilon - \theta_i) \frac{\partial f}{\partial \theta_i}(\theta) + \sum_{i \in V} \sum_{j \in V} (\hat{\theta}_i^\epsilon - \theta_i)(\hat{\theta}_j^\epsilon - \theta_j) \frac{\partial^2 f}{\partial \theta_i \partial \theta_j}(\theta^*), \tag{72}$$

for some θ^* in the closed interval bounded by θ_i and $\hat{\theta}_i^\epsilon$. Substituting in the formula for $\hat{\theta}^\epsilon$ and collecting terms in powers of ϵ we get

$$f(\hat{\theta}^\epsilon) = f(\theta) + \epsilon \sum_{i=1}^{m} \theta_i \left(\frac{\partial f}{\partial \theta_i}(\theta) \right)^2 + O(\epsilon^2). \tag{73}$$

The expression $\sum_{i \in V} \theta_i (\frac{\partial f}{\partial \theta_i}(\theta))^2$ is always strictly positive under the assumptions made in the theorem. The $O(\epsilon^2)$ term can therefore always be dominated for sufficiently small ϵ, thus proving (71). □

The proof of Theorem 6.1 tells us only that some suitable $\epsilon > 0$ exists, and does not provide us with an ϵ for which the theorem holds. Such a value may in fact be found using the theory developed in Gopalakrishnan, Kanevsky, Nädas and Nahamoo (1991). Unfortunately this value is of no practical use, as it is far too small to yield an efficient update rule. However, as we demonstrate in the next two sections, conducting a line search along the direction $\langle \theta_i (\partial f / \partial \theta_i) \rangle$ is both effective and efficient.

6.2. A numerical optimization strategy

We can improve (increase) our objective function f by choosing $\hat{\theta}_i^{\epsilon} = \theta_i + \epsilon \theta_i \frac{\partial f}{\partial \theta_i}(\theta)$, $i = 1, 2, \ldots, m$ for some value $\epsilon > 0$ unless we are at a stationary or boundary point. To satisfy the constraint $0 \leq \hat{\theta}_i^{\epsilon} \leq 1$ for $i = 1, 2, \ldots, m$ we also require that $\epsilon \leq \epsilon_{\max}$, where

$$\epsilon_{\max} = \max \left\{ \epsilon : 0 \leq \theta_i + \epsilon \theta_i \frac{\partial f}{\partial \theta_i}(\theta) \leq 1, \text{ for } i = 1, 2, \ldots, m \right\} \tag{74}$$

if $\theta_i \frac{\partial f}{\partial \theta_i}(\theta) \neq 0$ for some $i = 1, 2, \ldots, m$, and $\epsilon_{\max} = 0$ otherwise.

Let us write $s(\theta)$ for our objective function $1 - S_A(P_\theta, \mathcal{C}, \mathcal{A})$. To optimize $s(\theta)$ numerically we used a standard line maximization routine to find a local maximum ϵ^{opt} for $s(\hat{\theta}^{\epsilon})$, subject to $0 \leq \epsilon^{\text{opt}} \leq \epsilon_{\max}$. We then repeated the procedure iteratively making $\hat{\theta}^{\epsilon}$ the new value of θ at each consecutive iteration.

In our numerical experiments we tried the golden section search method as well as Brent's method using derivatives. We used slightly modified versions of the implementations described in Press *et al.* (1999, pp. 397–408), of these two algorithms. As the performance of the two methods was very close, we chose to use the more robust golden section search method in our more extensive experiments. Also the routine for initially bracketing a minimum described in Press *et al.* (1999) was modified to account for the additional knowledge, $0 \leq \epsilon \leq \epsilon_{\max}$. We iterate the line maximization procedure until we achieve a satisfactory accuracy for θ or until a very large number of iterations have been made.

6.3. Computational efficiency

We now explain an efficient scheme for organizing the required arithmetic for the line search. Let each word in the vocabulary V be assigned an index, and define the matrices A and B as follows: $[A]_{ii} = p(a(w_i) \mid w_i\, h)$, with 0s elsewhere, and $[B]_{ij} = p(a(w_i) \mid w_j\, h)$, where in both cases the history is taken to be fixed. These definitions permit the numerator of Equation (67) to be expressed as the w-row of the matrix–vector product $A\theta$, and the denominator to be expressed as the w-row of the matrix–vector product $B\theta$.

A and B each have dimension $m \times m$, which for a typical IBM speech recognition vocabulary would be $70\,000 \times 70\,000$. In other words A and B have $70\,000^2 = 4{\cdot}9 \times 10^9$ elements. In reality A is stored as a diagonal matrix and B as a sparse matrix containing approximately 10^8 elements. But this still means that the computation of $s(\theta)$ requires at least 10^8 multiplications. Repeated evaluation of $s(\theta)$ is going to be very costly, and optimizing this function will be infeasible, unless we come up with some computational savings.

The most important saving comes from the observation that

$$s(\theta + \epsilon \nu) = \sum_{i=1}^{m} c_i \frac{(A\theta)_i + \epsilon (A\nu)_i}{(B\theta)_i + \epsilon (B\nu)_i}, \tag{75}$$

where $\nu = (\nu_1, \nu_2, \ldots, \nu_m)$, with $\nu_i = \theta_i \frac{\partial s}{\partial \theta_i}$, and hence $\hat{\theta}^{\epsilon} = \theta + \epsilon \nu$.

If we precompute $\alpha = A\theta$, $\beta = A\nu$, $\gamma = B\theta$ and $\delta = B\nu$ then the cost of evaluating $s(\hat{\theta}^{\epsilon})$ for a particular value of ϵ is m divisions, $3m$ additions and $3m$ multiplications.

Rewriting formula (75) in terms of α, β, γ and δ we get the expression

$$s(\hat{\theta}^{\epsilon}) = \sum_{i=1}^{N} c_i \frac{\alpha_i + \epsilon\beta_i}{\gamma_i + \epsilon\delta_i}. \tag{76}$$

Now observe that this may be written as

$$\begin{aligned} s(\hat{\theta}^{\epsilon}) &= \sum_{i=1}^{N} c_i \frac{\alpha_i}{\gamma_i} + \sum_{i=1}^{N} c_i \frac{\epsilon(\beta_i - \alpha_i d_i)}{\gamma_i + \epsilon\delta_i} \\ &= s(\theta) + \epsilon \sum_{i=1}^{N} \frac{f_i}{1 + \epsilon d_i}, \end{aligned} \tag{77}$$

where $d_i = \frac{\delta_i}{\gamma_i}$ and $f_i = c_i(\frac{\beta_i - \alpha_i d_i}{\gamma_i})$ for $i = 1, 2, \ldots, m$. After precomputing α, β, γ, δ, $d = (d_1, d_2, \ldots, d_m)$ and $f = (f_1, f_2, \ldots, f_m)$, the cost of evaluating $s(\hat{\theta}^{\epsilon})$ for a particular value of ϵ according to (77) is m divisions, $m + 1$ multiplications and $2m + 1$ additions. This is a total of $4m + 2$ arithmetic operations. Evaluating $s(\theta)$ for an arbitrary value of θ costs approximately 10^8 arithmetic operations and evaluating $s(\hat{\theta}^{\epsilon})$ for a particular ϵ costs approximately $4m = 4 \times 70\,000 = 2{\cdot}8 \times 10^5$ operations after we have performed all the precomputation. This means that we can evaluate $s(\hat{\theta}^{\epsilon})$ for more than 350 different values of ϵ for the cost of one general function evaluation once we have precomputed α, β, γ, δ, d and f! The precomputation of α, β, γ, δ, d and f costs roughly as much as two general function evaluations of $s(\theta)$. But we can cut this precomputation cost in half by the observation that once we find a best choice of ϵ, ϵ^{opt}, we can compute the next values for α and γ by the formula

$$\begin{bmatrix} \alpha \\ \gamma \end{bmatrix} = \begin{bmatrix} \alpha^{\text{old}} \\ \gamma^{\text{old}} \end{bmatrix} + \epsilon^{\text{opt}} \begin{bmatrix} \beta^{\text{old}} \\ \delta^{\text{old}} \end{bmatrix}. \tag{78}$$

The only costly precomputation step left is then to recompute β and δ for each new choice of θ.

In an efficient implementation of the optimization of $s(\theta)$ we can therefore find the best ϵ for the cost of only slightly more than one general function evaluation per consecutive line maximization if m is as large as 70 000.

7. Decoding results

We now report decoding results for various models trained by the SAWER objective function. By smoothing such models with others trained by the lexical perplexity criterion, we were able to obtain small reductions in word error rate. As we explain later, we achieved this result with no increase in the number of model parameters.

We experimented with a total of five language models (no relation to those in Section 5.3). No model made any use of history; we made this choice to keep the SAWER training computation tractable. We began with two base language models, the uniform model $p_0(w)$ and the unigram model $p_1(w)$, respectively defined as $p_0(w) = 1/|V|$ and $p_1(w) = c(w)/N$, where $c(w)$ is the count of w in the training corpus, and N is its size. The counts used for p_1 were determined from the same corpus of 1.0 GW discussed in Section 5.3.

TABLE III. WER results for base (p_0, p_1), SAWER (s_0, s_1) and mixture ($\bar{p}$) models. The parenthesized numbers are perplexities

	SOB		NRR		SPT		ALL	
p_0	78·8%	(6079)	76·2%	(5841)	87·4%	(1002)	84·9%	(1464)
p_1	40·9%	(1292)	47·3%	(1312)	59·7%	(463)	55·5%	(577)
s_0	85·0%	(6063)	84·4%	(6213)	90·1%	(2.5×10^6)	88·7%	(1.1×10^6)
s_1	51·1%	(4472)	58·3%	(3699)	66·1%	(1779)	63·0%	(2125)
$\bar{p}$	40·5%	(1540)	44·8%	(1461)	59·8%	(606)	55·2%	(735)

We used p_0 and p_1 as starting points for two models, respectively s_0 and s_1, trained by iterative improvement of the SAWER objective function, using the methods of the preceding section. The training data for s_0 was a synthetic corpus in which each word $w \in V$ occurred exactly once. The training data for s_1 was likewise synthetic, in which each word $w \in V$ appeared exactly $c(w)$ times. Thus s_0 and s_1 had access to no additional lexical data, other than that available to p_0 and p_1, respectively. The fifth model we experimented with was the linear mixture $\bar{p} = 0{\cdot}1s_0 + 0{\cdot}1p_0 + 0{\cdot}4s_1 + 0{\cdot}4p_1$. The mixture weights were picked *a priori* to indicate how important we felt each model should be relative to the other models. No tuning or other sets of weights were tried.

Our decoding experiments consisted of decoding the three test corpora used to create Figure 5. Our decoder had a vocabulary of 63 389 words, and used acoustic models trained on about 250 h of acoustic data. The results are summarized in Table III. We observe a small improvement in the word error rate for the mixture model. Note that this model is formally a list of $|V|$ unigram probabilities, and thus contains exactly the same number of parameters as p_1 (or s_1). Thus we have improved performance, over either p_1 or s_1 individually, without increasing the model size. It is not clear that these results are statistically significant. It should also be noted that SAWER alone degraded performance.

8. Additional applications

8.1. Vocabulary selection

Consider a corpus $\mathcal{C}$ and a given recognizer vocabulary V. Suppose we have a set of "unknown" words U that appear in $\mathcal{C}$ but are not present in V. We want to determine which $u \in U$, if any, to add to V.

First note why this is a problem. Augmenting V with some particular u will increase (from 0) the probability that we will decode this word correctly when we encounter it. But it also increases the probability that we will make errors on other words, since there is now a new way to make a mistake.

We proceed to estimate the change in error rate that follows from adding any given $u \in U$ to V. By the arguments given earlier, the synthetic acoustic word error rate on $\mathcal{C}$ is

$$S_{AV}(\mathcal{C}) = \frac{1}{N}\sum_{i\in\mathcal{C}}(1 - p_V(w_i \mid a(w_i)\, h_i)), \tag{79}$$

where p_V denotes computation of confusabilities with respect to the unaugmented vocabulary V. We assume that $p_V(w \mid a(w)\, h) = 0$ when $w \notin V$.

Suppose now that we form an augmented vocabulary $V' = V \cup \{u\}$. Then we recompute the synthetic acoustic word error rate as

$$S_{AV'}(\mathcal{C}) = \frac{1}{N}\sum_{i\in\mathcal{C}}(1 - p_{V'}(w_i \mid a(w_i)\, h_i))\cdot \tag{80}$$

We would hope that $S_{AV'}(\mathcal{C}) < S_{AV}(\mathcal{C})$, in other words that adjoining u causes the error rate to drop. Thus we define Δ_u, the *improvement* due to u, as

$$\Delta_u(\mathcal{C}) = S_{AV}(\mathcal{C}) - S_{AV'}(\mathcal{C}) = \frac{1}{N}\sum_{i\in\mathcal{C}}(p_{V'}(w_i \mid a(w_i)\,h_i) - p_V(w_i \mid a(w_i)\,h_i))\cdot \quad (81)$$

We propose to perform vocabulary selection by ranking the elements of U according to Δ_u, adjoining the element with the largest strictly positive improvement, and then repeating the computation.

8.2. Selection of trigrams and maxent features

This same general approach may be used for selecting features for maximum entropy models, based upon the acoustic perplexity or synthetic acoustic word error rate, or their analogs for a general channel (such as translation). In particular this applies to selecting trigrams (or higher order n-grams), for extending lower-order n-gram models, based upon the gain in acoustic perplexity or synthetic acoustic word error rate.

For example, in a way similar to the selection of words for vocabularies, we may ask what trigrams should be used to augment a base bigram language model. We will analyze this question in terms of the effect this augmentation would have on both the acoustic perplexity and synthetic acoustic word error rate.

We consider two language models: a base model $p(w \mid h)$ and an augmented model $p_{xyz}(w \mid h)$. Here the latter is obtained as a maximum entropy model, perturbing the base model according to

$$p_{xyz}(w \mid h) = \frac{p(w \mid h)\cdot e^{\lambda_{xyz} f_{xyz}(w,h)}}{Z(h,\lambda_{xyz})}\cdot \quad (82)$$

The feature indicator function $f_{xyz}(w,h)$ is triggered when $w = z$ and $h = xy$, and is defined by

$$f_{xyz}(w,h) = \begin{cases} 1 & \text{if } w = z \text{ and } h = xy \\ 0 & \text{otherwise.} \end{cases} \quad (83)$$

The exponent λ_{xyz} is determined in the usual maximum entropy manner (Rosenfeld, 1996) by the requirement

$$E_{p_{xyz}}[\mathcal{C}] = E_{\tilde{p}}[\mathcal{C}] \quad (84)$$

where $\tilde{p}$ is an empirical model of the corpus.

We want to know, what are the most valuable trigrams to use to augment the base model? We proceed to compute using the decoding probs determined with respect to these two different language models, respectively $p(w_i \mid a(w_i)\,h_i)$ and $p_{xyz}(w_i \mid a(w_i)\,h_i)$. We define the gain, which measures value according to acoustic peplexity, via

$$G_{xyz} = \frac{1}{N}\log\frac{P_{xyz}(\mathcal{C} \mid \mathcal{A})}{P(\mathcal{C} \mid \mathcal{A})}\cdot \quad (85)$$

Likewise we define the improvement, which measures value according to synthetic acoustic word error rate, via

$$\Delta_{xyz} = \frac{1}{N}(P_{xyz}(\mathcal{C} \mid \mathcal{A}) - P(\mathcal{C} \mid \mathcal{A}))\cdot \quad (86)$$

Both expressions are valid, and experimental methods can be used to determine which measure is appropriate to a particular task.

This approach can be generalized, when acoustic confusabilities can be determined for a particular speaker or acoustic environment. For instance, it would be possible to train a speaker-dependent language model, based upon the speaker's acoustics. Or likewise to train a language model, based upon a particular acoustic channel, like a bandwidth-limited telephone channel, or the cabin of an automobile. Indeed the same approach might be used to train a language model for a source-channel (statistical) machine translation system, which is sensitive to the characteristics of the translation channel.

9. Summary and review

In this paper we explored the theory and practice of acoustic confusability. We defined and motivated the statistics *acoustic perplexity* (Y_A) and *synthetic acoustic word error rate* (S_A). We showed how these depend upon the *acoustic encoding probability* $p(a(w) \mid x\, h)$, introduced a technique for computing this quantity synthetically, and explained how this computation may be efficiently implemented. We demonstrated the superiority of Y_A and S_A as predictors of language model performance, showed how these statistics may be used as objective functions for language model training, and presented a practical method for training a language model with synthetic acoustic word error rate as the objective function. We presented results from a simple speech recognition experiment, showing that a mixture model that includes a component trained with S_A as an objective function can yield a small reduction in word error rate (however, this result is not statistically significant). Moreover there are limitations to training language models using these methods: they are computationally costly and may not be practical for the construction of bigram or higher-order models.

We are extending the ideas of acoustic perplexity and synthetic acoustic word error rate to more sophisticated recognition experiments, and to such diverse tasks as vocabulary selection for speech recognition systems, and the ranking of features for maximum entropy language models.

References

Anton, H. (1973). *Elementary Linear Algebra*. John Wiley & Sons, New York, NY.

Apostol, T. M. (1957). *Mathematical Analysis*. Addison-Wesley, Reading, MA.

Bahl, L. R., Baker, J. K., Jelinek, F. & Mercer, R. L. (1977). Perplexity—a measure of the difficulty of speech recognition tasks. *Journal of the Acoustical Society of America*, **62 (Suppl 1)**, S63.

Bahl, L. R., Jelinek, F. & Mercer, R. L. (1983). A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-5**, 179–190.

Berger, A., Brown, P., Della Pietra, S., Della Pietra, V., Gillett, J., Lafferty, J., Printz, H. & Ureš, L. (1994). The Candide system for machine translation. *Proceedings of the DARPA Conference on Human Language Technology*, pp. 157–162.

Author: Please provide conference location and volume number for Berger et al. *(1994)*

Chen, S., Beeferman, D. & Rosenfeld, R. (1998). Evaluation metrics for language models. *Proceedings of the Broadcast News Transcription and Understanding Workshop*, Lansdowne, Virginia, pp. 275–280.

Chung, K. L. (1979). *Elementary Probability Theory with Stochastic Processes*. Springer-Verlag, New York, NY.

Clarkson, P. & Robinson, T. (1998). The applicability of adaptive language modelling for the broadcast news task. *Proceedings of the Fifth International Conference on Spoken Language Processing*, Sydney, Australia.

Cover, T. M. & Thomas, J. A. (1991). *Elements of Information Theory*. John Wiley and Sons, New York, NY.

Evermann, G. & Woodland, P. (2000). Large vocabulary decoding and confidence estimation using word posterior probabilities. *Proceedings of International Conference on Acoustics, Speech and Signal Processing'00*, Istanbul, Turkey, volume 3, pp. 2366–2369.

Ferretti, M., Maltese, G. & Scarci, S. (1990). Measuring information provided by language model and acoustic model in probabilistic speech recognition: theory and experimental results. *Speech Communication*, **9**, 531–539.

Gopalakrishnan, P. S., Kanevsky, D., Nádas, A. & Nahamoo, D. (1991). An inequality for rational functions with applications to some statistical estimation problems. *IEEE Transactions on Information Theory*, **37**, 107–113.

Herstein, I. N. (1975). *Topics in Algebra*, 2nd edition. John Wiley and Sons, New York, NY.

Hoel, P. G. (1984). *Introduction to Mathematical Statistics*, 5th edition. John Wiley and Sons, New York, NY.

Ito, A., Kohda, M. & Ostendorf, M. (1999). A new metric for stochastic language model evaluation. *Proceedings of the Sixth European Conference on Speech Communication and Technology*, Budapest, Hungary, volume 4, pp. 1591–1594.

Iyer, R., Ostendorf, M. & Meteer, M. (1997). Analyzing and predicting language model improvements. *IEEE Workshop on Automatic Speech Recognition and Understanding*.

Jelinek, F. (1997). *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, MA.

Klakow, D. & Peters, J. Testing the correlation of word error rate and perplexity. *Speech Communications*, to appear.

Kreyszig, E. (1978). *Introductory Functional Analysis with Applications*. John Wiley and Sons, New York, NY.

Mangu, L., Brill, E. & Stolcke, A. (1999). Finding consensus among words: Lattice-based word error minimization. *Proceedings of European Conference on Speech Communication and Technology'99*, Budapest, Hungary, pp. 495–498.

McAllaster, D. & Gillick, L. (1999). Studies in acoustic training and language modeling using simulated speech data. *Proceedings of the Sixth European Conference on Speech Communication and Technology*, Budapest, Hungary, volume 4, pp. 1787–1790.

McAllaster, D., Gillick, L., Scattone, F. & Newman, M. (1998). Fabricating conversational speech data with acoustic models: A program to examine model–data mismatch. *Proceedings of International Conference on Spoken Language Processing*, Sydney, Australia, pp. 986.

Press, W. H., Teukolsky, S. A., Vetterling, W. T. & Flannery, B. P. (1999). *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 2nd edition.

Rosenfeld, R. (1996). A maximum entropy approach to adaptive statistical language modeling. *Computer Speech and Language*, **10**, 187–228.

Wessel, F., Macherey, K. & Schlüter, R. (1998). Using word probabilities as confidence measures. *Proceedings of International Conference on Acoustics, Speech and Signal Processing'98*, Seattle, Washington, volume 1, pp. 225–228.

(Received xx Month xxxx and accepted for publication xx Month xxxx)