# Language modeling for speech recognition[1]

**Frederick Jelinek**
Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD 21218
jelinek@jhu.edu

## 1 Introduction

A speech recognizer is a device which automatically transcribes speech into text. It can be thought of as a voice actuated "typewriter" in which the transcription is carried out by a computer program and the transcribed text appears on a workstation display. The recognizer is usually based on some finite vocabulary that restricts the words that can be "printed" out. Until we state otherwise, the designation *word* denotes a *word form* defined by its spelling. Two differently spelled inflections or derivations of the same stem are considered different words (e.g., *table* and *tables)*. Homographs having different parts of speech (e.g., *absent* [VERB] and *absent* [ADJECTIVE]) or meanings (e.g., *bank* [FINANCIAL INSTITUTION] and *bank* [OF A RIVER]) constitute the same word.

## 2 A mathematical formulation

A succinct mathematical formulation of the speech recognition problem is as follows [2] [10].

Let $\mathbf{A}$ denote the acoustic evidence (data) on the basis of which the recognizer will make its decision about which words were spoken. Since we are dealing with digital computers, then without loss of generality we may assume that $\mathbf{A}$ is a sequence of symbols taken from some (possibly very large) alphabet $\mathcal{A}$:

$$\mathbf{A} = a_1 a_2 ... a_m \qquad a_i \in \mathcal{A} \tag{1}$$

The symbols $a_i$ can be thought of as having been generated in time, according to the index $i$.

Let

$$\mathbf{W} = w_1, w_2, ... w_n \qquad w_i \in \mathcal{V} \tag{2}$$

denote a string of $n$ words, each belonging to a fixed and known vocabulary $\mathcal{V}$.

If $P(\mathbf{W}|\mathbf{A})$ denotes the probability that the words $\mathbf{W}$ were spoken, given that the evidence $\mathbf{A}$ was observed, then the recognizer should decide in favor of a word string $\widehat{\mathbf{W}}$ satisfying

$$\widehat{\mathbf{W}} = \arg\max_{\mathbf{W}} P(\mathbf{W}|\mathbf{A}) \tag{3}$$

That is, the recognizer will pick the most likely word string given the observed acoustic evidence.

The well known Bayes' formula allows us to re-write the right-hand side probability of (3) as

$$P(\mathbf{W}|\mathbf{A}) = \frac{P(\mathbf{W})P(\mathbf{A}|\mathbf{W})}{P(\mathbf{A})} \tag{4}$$

where $P(\mathbf{W})$ is the probability that the word string $\mathbf{W}$ will be uttered, $P(\mathbf{A}|\mathbf{W})$ is the probability that when the speaker says $\mathbf{W}$ the acoustic evidence $\mathbf{A}$ will be observed, and $P(\mathbf{A})$ is the average probability that $\mathbf{A}$ will be observed. That is,

$$P(\mathbf{A}) = \sum_{\mathbf{W}'} P(\mathbf{W}')P(\mathbf{A}|\mathbf{W}') \tag{5}$$

Since the maximization in (3) is carried out with the variable $\mathbf{A}$ fixed (there is no other acoustic data save the one we are given), it follows from (3) and (4) that the aim of the recognizer is to find the word string $\widehat{\mathbf{W}}$ that maximizes the product $P(\mathbf{W})P(\mathbf{A}|\mathbf{W})$, i.e. satisfies

$$\widehat{\mathbf{W}} = \arg\max_{\mathbf{W}} P(\mathbf{W})P(\mathbf{A}|\mathbf{W}) \tag{6}$$

## 3 Components of a speech recognizer

Formula (6) determines what processes and components are of concern in the design of a speech recognizer.

### 3.1 Acoustic processing

First, it is necessary to decide what acoustic data $\mathbf{A}$ will be observed. That is, one needs to decide on a *"front end"* that will transform the pressure waveform (which is what sound is) into the symbols $a_i$ the recognizer will deal with. So, in principle, this front end includes a microphone whose output is an electric signal, a means of sampling that signal, and a manner of processing the resulting sequence of samples.

### 3.2 Acoustic modeling

Next, the recognizer needs to be able to determine the value $P(\mathbf{A}|\mathbf{W})$ of the probability that when the speaker uttered the word sequence $\mathbf{W}$ the acoustic processor produced the data $\mathbf{A}$. Since this number must be made available for all possible pairings of $\mathbf{W}$ with $\mathbf{A}$, it follows that it must be *computable "on the fly"*. The number of different possible values of $\mathbf{A}$ and $\mathbf{W}$ is just too large to permit a look-up.

[1] Portions of this article will be found in extended form in various chapters of a text to be published by MIT Press.


Proceedings of the ECAI 96 Workshop
*Extended Finite State Models of Language*
Edited by A. Kornai.

Thus to compute $P(\mathbf{A}|\mathbf{W})$ we need a statistical *acoustic model* of the interaction of the speaker with the acoustic processor. The total process we are modeling involves the way the speaker *pronounces* the words of $\mathbf{W}$, the ambience (room noise, reverberation, etc.), the microphone placement and characteristics, and the acoustic processing.

### 3.3 Language modeling

Formula (6) further requires that we be able to compute for every word string $\mathbf{W}$ the a priori probability $P(\mathbf{W})$ that the speaker wishes to utter $\mathbf{W}$. Bayes formula allows many decompositions of $P(\mathbf{W})$, but since the recognizer "naturally" wishes to convey the text in the sequence in which it was spoken, we will use the decomposition

$$P(\mathbf{W}) = \prod_{i=1}^{n} P(w_i|w_1, ..., w_{i-1}) \tag{7}$$

The recognizer must thus be able to determine estimates of the probabilities $P(w_i|w_1, ..., w_{i-1})$. We use the term *estimate* on purpose, because even for moderate values of $i$ and reasonable vocabularies, the probability $P(w_i|w_1, ..., w_{i-1})$ has just too many arguments. In fact, if $|\mathcal{V}|$ denotes the size of the vocabulary, then for $|\mathcal{V}| = 20,000$ and $i = 3$, the number of arguments is $8{\times}10^{12}$!

### 3.4 Hypothesis search

Finally, in order to find the desired transcription $\hat{\mathbf{W}}$ of the acoustic data $\mathbf{A}$ by formula (6), we must search over all possible word strings $\mathbf{W}$ to find the maximizing one. This search cannot be conducted by brute force: the space of $\mathbf{W}$s is astronomically large.

What is needed is a parsimonious hypothesis search that will not even consider the overwhelming number of possible candidates $\mathbf{W}$, and will only examine those word strings that are in some way suggested by the acoustics $\mathbf{A}$.

## 4 The language model problem

The task of the language model is to estimate the probabilities $P(w_i|w_1, ..., w_{i-1})$. It is, of course, absurd to think that the speaker's choice of his $i^{th}$ word actually depends on the entire *history* $\mathbf{h}_i \doteq w_1, ..., w_{i-1}$ of his previous speech. It is therefore reasonable that for purposes of the choice of $w_i$, the history be put into *equivalence classes* $\Phi(\mathbf{h}_i)$, where $\Phi$ is a many-to-one mapping of histories into some number $M$ of categories. Thus in reality formula (7) becomes

$$P(\mathbf{W}) = \prod_{i=1}^{n} P(w_i|\Phi(\mathbf{h}_i)) \tag{8}$$

and the art of *language modeling* consists of determining the appropriate equivalence classification $\Phi$ and the estimation of the probabilities $P(w_i|\Phi(\mathbf{h}_i))$.

One way the classifier might function is on the basis of a finite state "grammar". At time $i-1$, the grammar is in state $\Phi_{i-1}$, and the next word forces a change to state $\Phi_i$. Then (8) can be re-written as

$$P(\mathbf{W}) = \prod_{i=1}^{n} P(w_i|\Phi_{i-1}) \tag{9}$$

For this relatively simple situation, how would the probabilities $P(w_i|\Phi_{i-1})$ be estimated? One could acquire some large corpus of text of the kind that will be produced by the recognizer (e.g., if the application is the creation of medical reports, then this *training text* should consist of medical reports composed by any available method). One would then run the text's word sequences through the finite state grammar, and accumulate counts $C(w, \Phi)$ of the number of times the word $w$ was fed to the grammar immediately after the grammar was in state $\Phi$. If $C(\Phi)$ denotes the number of times the grammar reached state $\Phi$,

$$C(\Phi) = \sum_{w} C(w, \Phi) \tag{10}$$

then the simplest estimate of the desired probability would be

$$P(w_i|\Phi_i = \Phi) = \frac{C(w_i, \Phi)}{C(\Phi)} \tag{11}$$

Whatever the selected classification scheme, it will of necessity be a compromise between two requirements:

1. The classification must be sufficiently refined to provide adequate information about the history $\mathbf{h}$ so it can serve as a basis for prediction.
2. When applied to histories of the given training corpus, it must yield its $M$ possible classes frequently enough so that the probabilities $P(w|\Phi)$ can be reliably estimated (not necessarily by the crude relative frequency approach (11)).

The purpose of a language model for speech recognition is not an exact analysis for meaning extraction, but an apportionment of probability among alternative futures.

In the remainder of this article we will outline three basic methods of language model construction: trigram, decision tree, and maximum entropy. They are all finite state. The first, surprisingly powerful, is the standard approach. The remaining two are still experimental.

## 5 The trigram language model

The language model that is almost exclusively used by present day speech recognizers is based on a very simple equivalence classification: *histories are equivalent if they end in the same two words*. For such a *trigram* model,

$$P(\mathbf{W}) = \prod_{i=1}^{n} P(w_i|w_{i-2}, w_{i-1}) \tag{12}$$

Estimating the basic trigram probabilities by formulas (10) and (11) gives

$$P(w_3|w_1, w_2) = f(w_3|w_1, w_2) \doteq \frac{C(w_1, w_2, w_3)}{C(w_1, w_2)} \tag{13}$$

where $f(\,|\,)$ denotes the relative frequency function.

Unfortunately, (13) is a very inadequate formula, since many possible English word trigrams $w_1, w_2, w_3$ are never actually found even in very large corpora of training text. A language model based on (13) would assert the impossibility of such trigrams and would assign $P(\mathbf{W}) = 0$ to strings $\mathbf{W}$ containing them. A recognizer operating under the statistical decision

criterion (6) would thus be forced to commit a large number of errors. In fact, in the 1970ies an experiment was carried out at IBM in which a corpus of patent descriptions based on a 1000 word vocabulary was divided into test and training subsets (of size 300,000 and 1,500,000 words, respectively). It was found that 23% of the trigrams appearing in the test subset never took place in the training subset. Thus a language model based on formula (13) would have caused the recognizer to transcribe erroneously at least 23% of the spoken words.

It is therefore necessary to *smooth* the trigram frequencies. This can be done most simply by interpolating trigram, bigram, and unigram relative frequencies,

$$P(w_3|w_1,w_2) = \lambda_3 f(w_3|w_1,w_2) + \lambda_2 f(w_3|w_2) + \lambda_1 f(w_3) \tag{14}$$

where the non-negative weights satisfy $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

The smoothing weights are chosen by the maximum likelihood principle applied to *heldout* data. That is, the training text is divided into two parts: development and heldout. The relative frequencies $f(w_3|w_1,w_2)$, $f(w_3|w_2)$, and $f(w_3)$ are computed over the development data. Then the weights $\lambda_1, \lambda_2$, and $\lambda_3$ are adjusted so as to maximize the probability of the heldout text $\mathbf{W}^* = w_1^*, w_2^*, ..., w_N^*$ :

$$P(\mathbf{W}^*) = \prod_{i=1}^{N} P(w_i^*|w_{i-2}^*, w_{i-1}^*)$$

where $P(w_i^*|w_{i-2}^*, w_{i-1}^*)$ is given by formula (14). Finally, the relative frequencies $f(w_3|w_1,w_2)$, $f(w_3|w_2)$, and $f(w_3)$ are recomputed over the *entire* training text.

In actual fact, the weights should vary with the confidence that we place in the probability approximation provided by the relative frequency that their multiply. So they should depend on the *count* on which $f$ is based. One proceeds iteratively as follows:

$$\begin{aligned} P(w_3|w_2) &= \mu(C(w_2))\, f(w_3|w_2) \\ &\quad + [1-\mu(C(w_2))]\, f(w_3) \qquad (15) \\ P(w_3|w_1,w_2) &= \nu(C(w_1,w_2))\, f(w_3|w_1,w_2) \\ &\quad + [1-\nu(C(w_1,w_2))]\, P(w_3|w_2) \qquad (16) \end{aligned}$$

where $C(\ )$ denotes the count function. The weights $\mu$ are evaluated first, thus determining the estimate $P(w_3|w_2)$ that is plugged into (16) for the determination of the weights $\nu$. Of course,

$$\begin{aligned} \lambda_1 &= [1-\mu]\,[1-\nu] \\ \lambda_2 &= \mu\,[1-\nu] \\ \lambda_3 &= \nu \end{aligned} \tag{17}$$

There exists also an alternative method for estimating trigram probabilities [11]. It is the *back-off* estimate

$$P(w_3|w_1,w_2) = \begin{cases} f(w_3|w_1,w_2) & \text{if } C(w_1,w_2,w_3) \geq K \\ \alpha\, f(w_3|w_2) & \text{if } C(w_1,w_2,w_3) < K \\ & \text{and } C(w_2,w_3) \geq L \\ \beta\, f(w_3) & \text{otherwise} \end{cases} \tag{18}$$

where the thresholds $M$ and $L$ are chosen by the designer, and the coefficients $\alpha$ and $\beta$ are needed so that $\sum_{w_3} P(w_3|w_1,w_2) = 1$. They are computed in conformity with an argumentation underlying the well-known Good-Turing estimates.[9]

## 6 Equivalence classification by decision trees

The equivalence classification of history $\mathbf{h}$ implicit in trigram language modeling was chosen arbitrarily. It is very mechanical. For instance, isn't it conceivable that in certain circumstances the word $w_{i-3}$ would have more influence than $w_{i-2}$ on the prediction of $w_i$? In principle, it would be desirable to determine the classification $\Phi(\mathbf{h})$ on the basis of some performance criterion applied to text data. Decision trees provide a means to our end.

In a tree language model the equivalence class $\Phi(\mathbf{h})$ is determined by successive answers to questions about the history $\mathbf{h}$ [1]. For instance, in the example of Figure 1, the classifier first asks the question $Q_{10}$ about $\mathbf{h}$. If the answer is *No* then the question $Q_{21}$ is posed. If the answer to $Q_{21}$ is *Yes*, then $Q_{32}$ is posed. If this answer is *Yes*, then $\mathbf{h}$ is classified as belonging to class $\Phi_{44}$; if the answer is *No* then $\mathbf{h}$ belongs to class $\Phi_{45}$. Note that the tree of Figure 1 is not uniformly developed. So if $Q_{20}$ is answered *No* then no further questions are asked. As a result, $\Phi_{42} = \Phi_{43} = \Phi_{31}$ and $\Phi_{46} = \Phi_{47} = \Phi_{33}$.

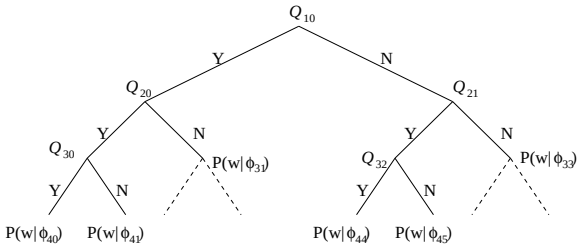


**Figure 1.**

How would one estimate the probabilities $P(w|\Phi_{4j})$ for the equivalence classes of the tree of Figure 1? Simply by taking a text training corpus, and creating word collections $\mathcal{S}_{4j}$ consisting of those words $w_i$ which in that corpus followed such histories $\mathbf{h}_i$, $i = 1, ..., n$, for which $\Phi(\mathbf{h}_i) = \Phi_{4j}$. The probability $P(w|\Phi_{4j})$ would then be determined according to the number of times the word $w$ was found in the collection $\mathcal{S}_{4j}$.[2]

In designing decision trees we are faced with three problems:

1. What *types* of questions should be asked of the history?
2. What *actual* questions should be asked at a particular stage of tree development?
3. Which question selection criterion should be used in the tree construction process?

In a very real sense, the first problem is hardest. There is an infinite number of possibilities. We must cut down on them. The conventional approach restricts questions about history $\mathbf{h}_i$ to questions about membership of words $w_{i-j}$, $j = 1, 2, ..., L$ in sets $\mathcal{R}$. Under this regime a potential question has the form "is $w_{i-j} \in \mathcal{R}$?".

Therefore, the actual questions asked depend on the choice of $j$ and of $\mathcal{R}$. In principle then, all combinations of $j$ and $\mathcal{R}$

[2] Strictly speaking, $P(w|\Phi_{4j})$ would be equal to the appropriately smoothed version of the relative frequency $f(w|\Phi_{4j})$.

can be evaluated by the selection criterion and the one giving the best result can be chosen. Since for a vocabulary of size $|\mathcal{V}|$ there are $\left(2^{|\mathcal{V}|}-1\right)/2$ different possible sets $\mathcal{R}$,[3] we must either find the desirable set by an algorithm or restrict drastically the number of sets that may be considered. Before discussing this problem further, we shall ask about the selection criterion.

A natural criterion is the probability of *unseen* text $\mathbf{W}^*$, as predicted by the decision tree language model, that is,

$$P(\mathbf{W}^*) = \prod_i P(w_i^*|\Phi(\mathbf{h}_i^*)) \tag{19}$$

where $\Phi$ is the equivalence classification induced by the tree. But to maximize (19) is the same as to minimize the *cross-entropy*

$$\begin{aligned} -\log P(\mathbf{W}^*) &= H_C \\ &\doteq -\sum_{j=1}^{M} C(\mathbf{h}^* \in \Phi_j) \sum_{v \in \mathcal{V}} C(v|\mathbf{h}^* \in \Phi_j) \log P(v|\mathbf{h}^* \in \Phi_j) \end{aligned} \tag{20}$$

where $\Phi_j$ are the different equivalence classes induced by the decision tree, $C(\mathbf{h}^* \in \Phi_j)$ is the number of times a history $\mathbf{h}^* \in \Phi_j$ appears in $\mathbf{W}^*$, $C(v|\mathbf{h}^* \in \Phi_j)$ is the number of times in $\mathbf{W}^*$ that $w_i = v$ when $\mathbf{h}_i^* \in \Phi_j$, and $P(v|\mathbf{h}^* \in \Phi_j)$ is the probability the language model assigns to the word $v$ when the history belongs to class $\Phi_j$.

Since our ignorance of $\mathbf{W}^*$ makes the minimization of (20) impossible, the standard approach is to try to minimize the same quantity over *training* data $\mathbf{W}$, or, equivalently,[4] minimize the empirical entropy

$$H_f \doteq -\sum_{j=1}^{M} f(\mathbf{h}^* \in \Phi_j) \sum_{v \in \mathcal{V}} f(v|\mathbf{h}^* \in \Phi_j) \log f(v|\mathbf{h}^* \in \Phi_j) \tag{21}$$

where $f$ again denotes the relative frequency.

We can now return to the choice of the sets $\mathcal{R}$ underlying the defining questions "is $w_{i-j} \in \mathcal{R}$?". There is an iterative algorithm due to Chou[6] which, starting with an initial guess at $\mathcal{R}$, gradually improves it, that is, obtains after every step a new $\mathcal{R}'$ that lowers the value of $H_f$ defined by (21). This would naturally lead to the following general step of decision tree development:

1. Given the tree developed so far, choose a leaf to be split.
2. For the chosen leaf and $j = 1, 2, ..., L$ use Chou's algorithm to find sets $\mathcal{R}_j$ leading to the "best" questions of the type "is $w_{i-j} \in \mathcal{R}_j$?".
3. Determine $j^* \in \{1, 2, ..., L\}$ for which the question "is $w_{i-j^*} \in \mathcal{R}_{j^*}$?" applied to the chosen leaf leads to the lowest value of $H_f$.
4. Use the question "is $w_{i-j^*} \in \mathcal{R}_{j^*}$?" to split the leaf.

Besides the fact that the derived sets $\mathcal{R}_{j^*}$ depend on the starting set and are not necessarily optimal, and that questions of the type "is $w_{i-j^*} \in \mathcal{R}_{j^*}$?" are not as general as one would desire, there are other very serious problems with the method, all connected with the necessary sparseness of available training data. First, at every step the entropy $H_f$ is lowered so the method has no built-in stopping rule. Second, the relative frequencies $f(v|\mathbf{h}^* \in \Phi_j)$ suffer from data fragmentation (the further is the split leaf from the tree root, the more sparse is the data) so the sets $\mathcal{R}$ are far from optimal for *unseen* data. And third, Chou's algorithm gives no guidance whether to include words $v$ in $\mathcal{R}$ or in its complement $\overline{\mathcal{R}}$ when $w_{i-j} \neq v$ for *all* values of $i$ in the training set.

While the stopping rule is routinely based on cross-validation [14],[5] there are no universally accepted methods to palliate the two other problems.

An alternative to the Chou procedure is due to Mercer and Brown[4]. It consists of limiting a priori the possible sets $\mathcal{R}$ to some fixed collection $\mathcal{U}$, and splitting a leaf by exhaustively searching for the best question "is $w_{i-j} \in \mathcal{R}$?" for $j \in \{1, 2, ..., L\}$ and $\mathcal{R} \in \mathcal{U}$. The point is not only that the question search is smaller, but that the sets $\mathcal{R} \in \mathcal{U}$ consist of words that have an essential relation to each other: it is expected that if the words $v \in \mathcal{R}$ that are found in the training data behave in a certain way, so will most other words in $\mathcal{R}$ when encountered in unseen data.

In the Mercer / Brown procedure the sets of $\mathcal{U}$ form a *nested* hierarchy which can be thought of as a tree whose leaves are words. The sets $\mathcal{R}$ then correspond to *nodes* in the tree, $\mathcal{R}$ consisting of those words which are found on the leaves of the subtree stemming from the node in question. The word trees are constructed by automatic clustering of text data.[5]

## 7 Equivalence classification of history under a maximum entropy regime

### 7.1 Introduction

In Section 5 we discussed two methods of trigram language model construction that combined differently information obtained from a training text: linear interpolation of separate "knowledge sources" of (14),

$$P(w_3|w_1, w_2) = \lambda_3 f(w_3|w_1, w_2) + \lambda_2 f(w_3|w_2) + \lambda_1 f(w_3)$$

and the back-off to these knowledge sources of (18),

$$P(w_3|w_1, w_2) = \begin{cases} f(w_3|w_1, w_2) & \text{if } C(w_1, w_2, w_3) \geq K \\ \alpha\, f(w_3|w_2) & \text{if } C(w_1, w_2, w_3) < K \\ & \quad \text{and } C(w_2, w_3) \geq L \\ \beta\, f(w_3) & \text{otherwise} \end{cases}$$

The first method weighs differently our confidence in the different information, while the second uses tests to choose the information that is considered reliable.

A closely related point of view (that combines information differently) is the following: Construct the joint probability $P(w, \mathbf{h})$ by insisting that

- $P(w, \mathbf{h})$ should satisfy certain linear constraints, and

[3] This is a very huge number for realistic vocabularies $\mathcal{V}$.

[4] The equivalence holds if we estimate the desired probabilities by relative frequency, that is, if we set $P(v|\mathbf{h} \in \Phi_j) = f(v|\mathbf{h} \in \Phi_j)$.

[5] That is, part of the training text is held out from the construction of the sets $\mathcal{R}$. The cross-entropy (20) is then evaluated over this heldout text. If it does not decrease after a split of any leaf, the split is nullified, and no further splitting of this leaf is carried out.

- $P(w, \mathbf{h})$ should organize itself in all other respects in accordance with our ignorance about everything not specified by these constraints. [6]

As an example,[7] we may wish to construct a trigram probability $P(w_1, w_2, w_3)$ having maximal entropy and satisfying the following constraints:

$$\begin{array}{ll} P(w_1,w_2,w_3) = f(w_1,w_2,w_3) & \text{if } C(w_1,w_2,w_3) \geq K \\ P(w_1,w_3) = f(w_1,w_3) & \text{if } C(w_1,w_3) \geq L \\ P(w_2,w_3) = f(w_2,w_3) & \text{if } C(w_2,w_3) \geq L \\ P(w_3) = f(w_3) & \text{if } C(w_3) \geq M \\ P(w_1,w_2) = f(w_1,w_2) & \text{if } C(w_1,w_2) \geq M \\ \sum_{w_1,w_2,w_3} P(w_1,w_2,w_3) = 1 & \end{array} \tag{22}$$

The idea of (22) is that when the training data satisfies the count conditions, the computed relative frequencies constitute a reliable estimate of the corresponding probabilities.

## 7.2 The general approach

The general approach is as follows ([8] and [7]): Let $\mathrm{x} = x_1, x_2, ..., x_n$ denote a sequence of $n$ random variables. Denote by $k(\mathrm{x}|i)$ the $i^{th}$ *constraint function* which need not be (but often is) an indicator function (as it would be in (22)).[8] Then

- Determine $P(\mathrm{x})$ so that
  - it satisfies
$$\sum_{\mathrm{x}} P(\mathrm{x}) k(\mathrm{x}|i) = d(i) \tag{23}$$
    for given *constraint targets* $d(i)$, $i = 1, 2, ..., m$, and
  - $P(\mathrm{x})$ has maximal entropy.

It is easy to prove that if the constraints (23) are consistent,[9] then the desired distribution has the parametric form

$$P(\mathrm{x}) = [\exp \lambda_0] \left[ \prod_{i=1}^{m} \exp\{\lambda_i k(\mathrm{x}|i)\} \right] \tag{24}$$

where the coefficients $\lambda_0, \lambda_1, ..., \lambda_m$ are determined so the constraints (23) are satisfied. Note that since $P(\mathrm{x})$ is a probability it must satisfy an additional, zeroth constraint

$$k(\mathrm{x}|0) = 1 \qquad \text{for all } \mathrm{x} \tag{25}$$

Hence the normalizing parameter $\lambda_0$.

[6] I.e., among all probabilites $P(w, \mathbf{h})$ satisfying the prescribed constraints, the one chosen should have maximum entropy

$$H = -\sum_{w,\mathbf{h}} P(w, \mathbf{h}) \log P(w, \mathbf{h})$$

[7] We will soon see that the method is much more general.

[8] An indicator function is one that takes on only two values, 0 and 1.

[9] That is, a probability distribution $P(\mathrm{x})$ actually exists satisfying them.

It is worth noting that if the constraint functions $k(\mathrm{x}|i)$ are indicator functions then the constraints (22) simply assure that the probability $P(\mathrm{x})$ have certain *prescribed* marginals.

Inspecting (24) we see that $P(\mathrm{x})$ is a product of factors, one for each constraint that the *particular* argument x is *involved in*. That is, if $k(\mathrm{x}|i) = 0$ for that value of x, then the $i^{th}$ factor is missing from the expression. This fact simplifies the use of $P(\mathrm{x})$ considerably.[10]

It is clear that the maximum entropy approach requires answers to the following two questions:

1. how to choose the constraints,
   and
2. how to solve for the parameters $\lambda_i$.

The choice of constraints is a difficult problem involving two aspects:

1. The choice of constraint functions $k(\mathrm{x}|j)$, i.e., of the *type* of desirable constraints.
2. The choice of constraint targets $d(j)$.

In choosing constraints, we must rely on intuition derived from our knowledge of the real underlying situation. There exists no generally applicable solution.

There are at least three methods for finding $\lambda_i$, all of which start with an initial guess:

1. hill climbing (*conjugate gradient descent*),
2. *alternating minimization* [7].
3. *iterative scaling* [8],

The last method is a particular version of the second which, although not the most efficient one, is in principle very simple and goes as follows:

1. Guess at the values of $\lambda_i$, $i = 1, 2, ..., m$.
2. For $j = 0$ to $m$, **do**:

   keeping $\lambda_i$, $i \neq j$ fixed, find $\lambda_j^*$ so as to satisfy the $j^{th}$ constraint;
   set $\lambda_j = \lambda_j^*$;
   **end**;
3. If all the constraints are sufficiently satisfied, then stop. Else go to 2.

The problem with alternating minimization is that its convergence may be slow, especially if the number of constraints $m$ is large. Particularly troublesome is the normalizing constraint (25) that apparently leads to a step involving a summation of $|\mathcal{V}|^n$ terms, where $|\mathcal{V}|$ is the size of the alphabet of $x_i$.

## 7.3 Simplification applicable to language modeling

The Della Pietras et al. [3] have made the important observation that if one is really interested in the conditional probability $P(w|\mathbf{h})$ (as we are) rather than in the joint $P(w, \mathbf{h})$,

[10] In the case of the constraints (22), there are potentially $5N+1$ constraints, where $N$ is the size of the training data. In fact, there are as many constraints of the first type as there are different trigrams that have been seen $K$ or more times in the training data.

then it is effective to limit computation by the rule:

$$\textit{use as one of the constraints}$$
$$P(\mathbf{h}) = \sum_{w} P(w,\mathbf{h}) = f(\mathbf{h}) \tag{26}$$

where $f$ denotes the relative frequency of events in the training set. Since *by definition* $\sum_{\mathbf{h}} f(\mathbf{h}) = 1$, this constraint assures the proper normalization of $P(w,\mathbf{h})$ and thus avoids the direct imposition of the normalizing constraint

$$\sum_{w,\mathbf{h}} P(w,\mathbf{h}) = 1$$

Furthermore, since $f(\mathbf{h})$ is non-zero for at most as many different histories $\mathbf{h}$ as there are words $N$ in the training data, carrying out the constraint (26) involves no more than $|\mathcal{V}|\times N$ operations, an amount which for reasonable size data sets and vocabularies is tolerable. In the next subsection we will analyze the computational problem when the last two trigram constraint types of (22) are replaced by the type $P(w_1,w_2) = f(w_1,w_2)$ analogous to (26). It will turn out that the upper bound $|\mathcal{V}|\times N$ applies to the computation of all constraint types.

It might seem that the constraint (26) may make it impossible for the probability $P(w|\mathbf{h})$ to be defined for histories $\mathbf{h}$ that were not seen in the training data (i.e. when $f(\mathbf{h}) = 0$). Fortunately, this is not so, as we will now show.

Because $P(w,\mathbf{h})$ is a product of factors (see (24)), one for each constraint, then

$$P(w,\mathbf{h}) = g(\mathbf{h})q(w,\mathbf{h})$$

where $g(\mathbf{h})$ is the factor arising from the constraint (26) and $q(w,\mathbf{h})$ denotes the product of the remaining factors. The required constraint (26) then forces

$$g(\mathbf{h}) = \frac{f(\mathbf{h})}{\sum_{w'} q(w',\mathbf{h})}$$

so that

$$P(w|\mathbf{h}) = \frac{P(w,\mathbf{h})}{P(\mathbf{h})} = \frac{g(\mathbf{h})q(w,\mathbf{h})}{f(\mathbf{h})} = \frac{q(w,\mathbf{h})}{q(w,\mathbf{h}) + \sum_{w'\neq w} q(w',\mathbf{h})}$$

will exist even for histories $\mathbf{h}$ that do not appear in the training corpus, unless $q(w,\mathbf{h}) = 0$. Even in the latter case $P(w|\mathbf{h})$ would be well defined except if $q(w',\mathbf{h}) = 0$ for *all* $w'$. But this could only be if for *every* $w'$ there existed at least one constraint involving the pair $w'$, $\mathbf{h}$ whose constraint target would equal 0.

## 7.4 Analysis of computational requirements

We will now see on the example of estimating $P(w_3|w_1,w_2)$ what the computational advantage of the constraint (26) amounts to. Suppose that we impose the constraints[11]

$$\begin{array}{ll} P(w_1,w_2,w_3) = f(w_1,w_2,w_3) & \text{if } C(w_1,w_2,w_3) \geq K \\ P(w_1,w_3) = f(w_1,w_3) & \text{if } C(w_1,w_3) \geq L \\ P(w_2,w_3) = f(w_2,w_3) & \text{if } C(w_2,w_3) \geq L \\ P(w_3) = f(w_3) & \text{if } C(w_3) \geq M \\ P(w_1,w_2) = f(w_1,w_2) & \text{for all } w_1,w_2 \end{array} \tag{27}$$

Then the solution has the form[12]

$$P(w_1,w_2,w_3) = g(w_1,w_2,w_3)^{k(w_1,w_2,w_3)} g_1(w_3)^{k_1(w_3)} g_2(w_1,w_2) g_3(w_1,w_3)^{k_3(w_1,w_3)} g_4(w_2,w_3)^{k_4(w_2,w_3)}$$

and alternating minimization of Section 7.2 involves the steps

1. Guess at initial values of $g_1, g_2, g_3$ and $g_4$.
2. For all $w_1,w_2,w_3$ such that $k(w_1,w_2,w_3) = 1$, set

$$g(w_1,w_2,w_3) = f(w_1,w_2,w_3)\left[g_1(w_3)^{k_1(w_3)} g_2(w_1,w_2) g_3(w_1,w_3)^{k_3(w_1,w_3)} g_4(w_2,w_3)^{k_4(w_2,w_3)}\right]^{-1}$$

3. For all $w_3$ such that $k_1(w_3) = 1$, set

$$g_1(w_3) = f(w_3)\left[\sum_{w_1,w_2} g(w_1,w_2,w_3)^{k(w_1,w_2,w_3)} g_2(w_1,w_2) g_3(w_1,w_3)^{k_3(w_1,w_3)} g_4(w_2,w_3)^{k_4(w_2,w_3)}\right]^{-1}$$

4. For all $w_1,w_2$, set

$$g_2(w_1,w_2) = f(w_1,w_2)\left[\sum_{w_3} g(w_1,w_2,w_3)^{k(w_1,w_2,w_3)} g_1(w_3)^{k_1(w_3)} g_3(w_1,w_3)^{k_3(w_1,w_3)} g_4(w_2,w_3)^{k_4(w_2,w_3)}\right]^{-1}$$

5. For all $w_1,w_3$ such that $k_3(w_1,w_3) = 1$, set

$$g_3(w_1,w_3) = f(w_1,w_3)\left[\sum_{w_2} g(w_1,w_2,w_3)^{k(w_1,w_2,w_3)} g_1(w_3)^{k_1(w_3)} g_2(w_1,w_2) g_4(w_2,w_3)^{k_4(w_2,w_3)}\right]^{-1}$$

[11] Compare with the constraints (22).

[12] In fact, consider the factors $\exp\{\lambda_i k(\mathbf{x}|i)\}$ in (24) corresponding to the first line of constraints of (27). The value of $\lambda_i$ is a function of the tripplet $w_1,w_2,w_3$ which is specified by the index $i$. So $\exp\lambda_i$ can be written as $g(w_1,w_2,w_3)$. Defining then the indicator function

$$k(w_1,w_2,w_3) \doteq \begin{cases} 1 & \text{if } C(w_1,w_2,w_3) \geq K \\ 0 & \text{otherwise} \end{cases}$$

we can write $\exp\{\lambda_i k(\mathbf{x}|i)\}$ as $g(w_1,w_2,w_3)^{k(w_1,w_2,w_3)}$. A similar change of notation can be applied to the other constraint types.

6. For all $w_2, w_3$ such that $k_4(w_2, w_3) = 1$, set

$$g_4(w_2, w_3) = f(w_2, w_3)\left[\sum_{w_1} g(w_1, w_2, w_3)^{k(w_1, w_2, w_3)} g_1(w_3)^{k_1(w_3)} g_2(w_1, w_2) g_3(w_1, w_3)^{k_3(w_1, w_3)}\right]^{-1}$$

7. Test to see if all constraints are now sufficiently satisfied. If they are, then stop; if they are not, go to step 2.

Since, because of step 4, $g_2(w_1, w_2) = 0$ whenever $f(w_1, w_2) = 0$, steps 3 and 4 require no more than $|\mathcal{V}| \times N$ additions. The same is true about steps 2, 5, and 6 because $f(w_1, w_2, w_3)$, $f(w_1, w_3)$, and $f(w_2, w_3)$, respectively, are non-zero for at most $N$ arguments. We see that the bound $|\mathcal{V}| \times N$ holds for each constraint type as long as the type (26) is imposed, regardless of the complexity or the number of constraint types.

## 7.5 Discussion

A trigram language model satisfying constraints (27) has actually been developed [12]. It performs somewhat better than the conventional language models (14) and (18). It has the added advantage that its size is scalable by adjustment of the thresholds $K, L$, and $M$.

Furthermore, the method allows the imposition of other than relative frequency constraints, some with linguistic motivation. For instance, one could constrain the values of the marginals

$$\sum_{w_1, w_2, w_3} P(w_1, w_2, w_3)\, k(w_1, w_2, w_3 | \pi_1, \pi_2, \pi_3)$$

where $\pi(w)$ denotes the part of speech of $w$, and

$$k(w_1, w_2, w_3 | \pi_1, \pi_2, \pi_3) \doteq \begin{cases} 1 & \text{if } \pi(w_i) = \pi_i, i = 1, 2, 3 \\ 0 & \text{otherwise} \end{cases}$$

A very successful application of the method concerns the inclusion of *triggers* in the determination of the equivalence class $\Phi(\mathbf{h})$ of the history [13]. These triggers are words present in the longer past which influence the distribution of the word being predicted.

When compared to the decision tree method, the maximum entropy approach has the advantage of not splitting data. Its main problem is the choice of constraints, the rigidity with which the targets $d(i)$ (see (23)) control the distribution, and the large computational effort necessary for determining the parameter values $\lambda_i$.

## REFERENCES


[1] L.R. Bahl, P.F. Brown, P.V. deSouza, and R.L. Mercer, 'A tree-based language model for natural language speech recognition', *IEEE Transactions on Acoustics, Speech and Signal Processing*, **37**, 1001–1008, (July 1989).

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

[3] A.L. Berger, S.A. Della Pietra, and V.J. Della Pietra, 'A maximum entropy approach to natural language processing', *Computational Linguistics*, **22**(1), (March 1996).

[4] P.F. Brown, S.A. Della Pietra, V.J. Della Pietra, R.L. Mercer, and P.S. Resnik. Language modeling using decision trees. An IBM Research Report, 1991.

[5] P.F. Brown, V.J. Della Pietra, P.V. deSouza, J.C. Lai, and R.L. Mercer, 'Class–based $n$–gram models of natural language', *Computational Linguistics*, **18**(4), 467–480, (December 1992).

[6] P.A. Chou, 'Optimal partitioning for classification and regression trees', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **13**(4), 340–354, (April 1991).

[7] I. Csiszar and G. Longo, 'Information geometry and alternating minimization procedures', *Statistics and Decisions*, **Supplement Issue 1**, 205–237, (1984).

[8] J.N. Darroch and D. Ratcliff, 'Generalized iterative scaling for log-linear models', *The Annals of Mathematical Statistics*, **43**, 1470–1480, (1972).

[9] I.J. Good, 'The population frequencies of species and the estimation of population parameters', *Biometrika*, **40**(parts 3 and 4), 237–264, (December 1953).

[10] F. Jelinek, L.R. Bahl, and R.L. Mercer, 'Design of a linguistic statistical decoder for the recognition of continuous speech', *IEEE Transactions on Acoustics, Speech and Signal Processing*, **IT-21**, 250–256, (1975).

[11] S. Katz, 'Estimation of probabilities from sparse data for the language model component of a speech recognizer', *IEEE Transactions on Acoustics, Speech and Signal Processing*, **35**(3), 400 – 401, (March 1987).

[12] R. Lau, R. Rosenfeld, and S. Roukos. Method for building scalable n-gram language models using maximum likelihood maximum entropy n-gram models. U.S. patent filed 1994.

[13] R. Rosenfeld, *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*, Ph.D. dissertation, Carnegie Mellon University, April 1994. Also published as Technical Report CMU-CS-94-138, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, April 1994.

[14] M. Stone, 'Cross–validatory choice and assessment of statistical predictions', *Journal of Royal Statistical Society*, **B 36**, 111–147, (1974).