ELSEVIER

Speech Communication 24 (1998) 171–192

SPEECH COMMUNICATION

A study of *n*-gram and decision tree letter language modeling methods

Gerasimos Potamianos [a,*], Frederick Jelinek [b]

[a] *Speech and Image Processing Services Research Lab, AT&T Labs-Research, 180 Park Avenue, Room D191, Florham Park, NJ 07932-0971, USA*

[b] *Center for Language and Speech Processing, The Johns Hopkins University, Baltimore, MD 21218, USA*

Received 11 August 1997; received in revised form 2 March 1998

Abstract

The goal of this paper is to investigate various language model smoothing techniques and decision tree based language model design algorithms. For this purpose, we build language models for printable characters (letters), based on the Brown corpus. We consider two classes of models for the text generation process: the *n*-gram language model and various decision tree based language models. In the first part of the paper, we compare the most popular smoothing algorithms applied to the former. We conclude that the bottom-up deleted interpolation algorithm performs the best in the task of *n*-gram letter language model smoothing, significantly outperforming the back-off smoothing technique for large values of *n*. In the second part of the paper, we consider various decision tree development algorithms. Among them, a *K*-means clustering type algorithm for the design of the decision tree questions gives the best results. However, the *n*-gram language model outperforms the decision tree language models for letter language modeling. We believe that this is due to the predictive nature of letter strings, which seems to be naturally modeled by *n*-grams.

Zusammenfassung

Das Ziel dieses Beitrags ist verschiedene Techniken zur Glättung von Sprachmodellen und Algorithmen zum Entwurf von Sprachmodellen auf der Basis von Entscheidungsbäumen zu untersuchen. Zu diesem Zweck verwenden wir den Brown-Korpus um Modelle von Buchstabenfolgen zu erstellen. Wir betrachten zwei Klassen von Modellen zur Textgenerierung: das *n*-Gramm Sprachmodell sowie verschiedene auf Entscheidungsbäumen basierende Verfahren. Im ersten Teil dieses Beitrags vergleichen wir die am häufigsten benutzten Glättungsalgorithmen angewandt auf *n*-Gramme. Wir folgern, daß der "bottom-up deleted interpolation"-Algorithmus am besten zur Glättung von *n*-Gramm Sprachmodellen geeignet ist und für große *n* dem "back-off"-Verfahren deutlich überlegen ist. Im zweiten Teil dieses Beitrags betrachten wir dann verschiedene Algorithmen zur Bildung von Entscheidungsbäumen. Unter diesen erzielt ein *K*-means-ähnlicher Algorithmus die besten Ergebnisse beim Entwurf der Fragen die der Entscheidungsbaum stellt. Für die Modellierung von Buchstabenfolgen erzielt das *n*-Gramm Sprachmodell aber trotzdem noch bessere Ergebnisse als alle Entscheidungsbäume. Wir glauben, daß dies durch die Fähigkeit der *n*-Gramme Buchstabenfolgen vorherzusagen begründet ist.

* Corresponding author. E-mail: makis@research.att.com.

0167-6393/98/$19.00
PII: S 0 1 6 7 - 6 3 9 3 (9 8) 0 0 0 1 8 - 1

Résumé

Le but de cet article est d'étudier différentes techniques de lissage de modèles de langage et différents algorithmes de construction de modèles de langage à base d'arbres de décision. Pour cela, nous construisons des modèles de langage pour des caractères écrits (lettres) à partir du Brown corpus. Nous considérons deux classes de modèles pour le processus de génération du texte: le modèle de langage n-gram, et différents modèles de langage à base d'arbres de décision. Dans la première partie de l'article, nous comparons les algorithmes de lissage les plus couramment appliqués au modèle de langage n-gram. L'algorithme "bottom-up deleted interpolation" donne les meilleurs résultats pour le lissage du modèle de langage n-gram, dépassant de fa c on significative la technique de lissage "back-off" pour de grandes valeurs de n. Dans la seconde partie de l'article, nous considérons différents algorithmes de développement d'arbres de décision. Parmi eux, un algorithme de type classification K-means aboutit aux meilleurs résultats pour la construction des arbres de décision. Cependant, le modèle de langage n-gram fournit de meilleurs résultats que les modèles de langage à arbres de décision pour la modélisation du langage écrit. Nous croyons que cela est dû à la nature prédictive des chaînes de caractères, qui semble être naturellement modélisée par les n-grams.

Keywords: Language modeling; n-grams; Decision trees; Smoothing; Laws of succession; Back-off language model; Deleted interpolation; Brown corpus

1. Introduction

Although less interesting than *word language models*, *letter* language models are suitable for certain applications, such as text compression (Bell et al., 1990), spelling letter automatic recognition (Betz and Hild, 1995), and estimation of language entropy (Brown et al., 1992a; Shannon, 1951). In addition, letter language models are simpler to construct, since their *vocabulary* size is significantly smaller than that of a typical word language model. In fact, their small vocabulary size allows a clear demonstration of the effects of *sparseness* of the training data on the language model performance. In an n-gram letter language model, data sparseness appears gradually, as n increases. In contrast, it abruptly arises in large vocabulary word language models, unless more sophisticated *class* language models are used (Brown et al., 1992b). Clearly, letter language models provide simple testing grounds of the language modeling techniques that constitute the subject of this paper. Such techniques are purely statistical methods, therefore, their generalization to word language modeling is straightforward.

The most widely adopted language model is the *n-gram* language model (Jelinek, 1997). Our motivation has been to outperform it by exploring *decision tree* (Breiman et al., 1984) based language models. Our results however are negative. Our decision tree based letter language models fail to improve the n-gram one, unless certain n-gram like constrains are imposed on the decision tree question design. Nevertheless, this paper contains a number of original results. First, although decision tree based language models have appeared before (Bahl et al., 1989), our work constitutes the first fair comparison between them and n-gram language models. In addition, the employment of the *K-means clustering* algorithm for decision tree question design (Chou, 1991) is its first application to language modeling. Finally, our decision tree *smoothing* algorithm is worth reporting. We propose the use of either *bottom–up deleted interpolation* (Jelinek and Mercer, 1980) or *back-off* smoothing (Katz, 1987), depending on the *depth* of each tree *leaf*. The investigation of suitable decision tree smoothing techniques has led us to a comparative study of the most popular smoothing techniques. Our conclusion is important. The bottom–up deleted interpolation smoothing is the most suitable smoothing technique for letter language models, being significantly more robust to sparseness of the training data than both *top–down* deleted interpolation and back-off smoothing.

Let $\boldsymbol{\omega}_1^m = \omega_1\omega_2\omega_3\ldots\omega_m$ be a sequence of m characters of text, where $\omega_i \in \upsilon$, for $i = 1, 2, 3, \ldots, m$, and υ denotes the vocabulary, which contains $|\upsilon|$ distinct characters. We consider the text generation process as a random one, therefore, $\boldsymbol{\omega}_1^m$ is drawn from an unknown probability mass function $\overline{\mathsf{Pr}}[\boldsymbol{\omega}_1^m] =$

$\prod_{i=1}^{m} \overline{\mathsf{Pr}}[\omega_i \mid \boldsymbol{\omega}_1^{i-1}]$. We are interested in letter language models that approximate these unknown conditional probability mass functions by *homogeneous* (i.e., independent of the character text location i) conditional probabilities

$$\overline{\mathsf{Pr}}[\omega_i \mid \boldsymbol{\omega}_1^{i-1}] \simeq \mathsf{Pr}[\omega_i \mid \boldsymbol{\Phi}(\boldsymbol{\omega}_{i-n+1}^{i-1})] = \mathsf{Pr}[\omega_0 \mid \boldsymbol{\Phi}(\omega_{-1}, \omega_{-2}, \ldots, \omega_{-n+1})] > 0, \tag{1}$$

for all $\boldsymbol{\omega}_{i-n+1}^{i} \in \upsilon^n$, and for a given positive integer n. In Eq. (1), $\boldsymbol{\Phi}$ denotes an *equivalence classification* on υ^{n-1}, whereas $\omega_{-j} \in \upsilon$ denotes the character lying j positions to the left of character $\omega_0 \in \upsilon$, that it helps to predict. Clearly, two issues are of interest in the design of language model (1). First, the specification of the equivalence classification $\boldsymbol{\Phi}$, and, second, the determination of probabilities (1). In this work, we consider n-gram and decision tree based language models, as a means of specifying $\boldsymbol{\Phi}$, and various smoothing techniques, as a means of determining the conditional probability mass functions (1).

In this case study for letter language modeling, all algorithms discussed are compared on the *Brown corpus* (Kucera and Francis, 1967), a popular corpus with the language modeling community (Brown et al., 1992a; Chen and Goodman, 1996; Ristad and Thomas, 1995, 1997a,b). More specifically, the Brown corpus, consisting of approximately 6.1 million characters with a vocabulary of 79 characters (see Table 2), is randomly divided into three sets, namely a *development*, a *held-out* (or, *check*), and a *test* set of approximately 4.9, 0.7, and 0.5 million characters, respectively [1] (the union of the development and held-out sets constitutes the training set). Counts $C(\omega_0, \omega_{-1}, \ldots, \omega_{-n+1})$, for all n-tuples $(\omega_0, \omega_{-1}, \ldots, \omega_{-n+1}) \in \boldsymbol{\upsilon}^n$, are collected from the three sets, denoted by $C_d(\bullet)$, $C_c(\bullet)$ and $C_t(\bullet)$, respectively. Given a language model (1), we define the *minus log-probability per character* on each of the three sets, denoted by $\mathscr{LP}_d$, $\mathscr{LP}_c$, and $\mathscr{LP}_t$, respectively, as

$$\mathscr{LP}_s = -\frac{\sum_{(\omega_0,\omega_{-1},\ldots,\omega_{-n+1})\in\upsilon^n} C_s(\omega_0, \omega_{-1}, \ldots, \omega_{-n+1}) \log_2 \mathsf{Pr}[\omega_0 \mid \boldsymbol{\Phi}(\omega_{-1}, \ldots, \omega_{-n+1})]}{\sum_{(\omega_0,\omega_{-1},\ldots,\omega_{-n+1})\in\upsilon^n} C_s(\omega_0, \omega_{-1}, \ldots, \omega_{-n+1})}, \tag{2}$$

for $s = d, c, t$, and measured in bits per character. Quantity (2) is consistently used throughout this paper, as a means of language model design and assessment. In more detail, counts $C_d(\bullet)$ are used to obtain the equivalence classification $\boldsymbol{\Phi}$ as well as, the *maximum likelihood* estimates of probabilities (1), by minimizing $\mathscr{LP}_d$. In addition, counts $C_c(\bullet)$ are used to optimize language model parameters, related to the smoothing of these maximum likelihood estimates, in order to improve the language model generalization to the held-out set. Minimization of $\mathscr{LP}_c$ is employed for this task. Finally, the performance of the resulting language models is assessed by means of $\mathscr{LP}_t$. With no loss of generality, alternatives to the held-out data scheme, described above, such as the *bootstrap*, the *leave-one-out*, and the *N-fold cross validation* methods (Efron, 1982; Ney et al., 1995) are not considered here. Such methods exhibit well known statistical advantages over the held-out data approach, however, they are significantly more computationally expensive.

This paper is structured as follows. In Section 2, various n-gram language model smoothing algorithms are discussed, namely various *laws of succession* [2] (Bell et al., 1990; Ney et al., 1995; Ristad, 1995), the *back-off* method (Katz, 1987; Ney et al., 1995), and *deleted interpolation* (Bahl et al., 1983, 1991; Chen and Goodman, 1996; Jelinek and Mercer, 1980; Ristad and Thomas, 1997b). Results on the Brown corpus are reported in Section 2.4. Section 3 is devoted to decision tree based language models. Two decision tree language model development algorithms are presented in Sections 3.2 and 3.3. Decision tree smoothing is discussed in Section 3.4, whereas, results on the Brown corpus are reported in Section 3.5. Section 4 summarizes our

[1] All held-out and test set characters appear in the development set, therefore, we have not introduced an *out of vocabulary* character in υ.

[2] We use this terminology to refer to smoothing techniques that take into account development set event occurrences only within the equivalence class $\boldsymbol{\Phi}(\omega_{-1}, \omega_{-2}, \ldots, \omega_{-n+1})$ under consideration.

conclusions. The reader is encouraged to consult our technical report for a more detailed version of this work (Potamianos and Jelinek, 1997). [3]

2. A comparison of various language model smoothing techniques for the n-gram language model

A conceptually simple but surprisingly powerful language model is the n-gram language model that corresponds to the equivalence classification (see also Eq. (1)) $\boldsymbol{\Phi}(\boldsymbol{h}_{n-1}) = \boldsymbol{h}_{n-1}$, where $\boldsymbol{h}_{n-1} = (\omega_{-1}, \omega_{-2}, \ldots, \omega_{-n+1})$ denotes the *history* of depth $n-1$, preceding character ω_0. We are interested in estimating $\mathsf{Pr}[\omega_0 = v \mid \boldsymbol{h}_{n-1}]$, for all $v \in \upsilon$, and all histories $\boldsymbol{h}_{n-1} \in \upsilon^{n-1}$. In practice, at least some such histories never occur in the development set. In these cases, we set

$$\mathsf{Pr}[v \mid \omega_{-1}, \omega_{-2}, \ldots, \omega_{-n+1}] = \mathsf{Pr}[v \mid \omega_{-1}, \omega_{-2}, \ldots, \omega_{-K}],$$

for all $v \in \upsilon$, where

$$K = \max_{1 \leqslant k < n} \{k: C_d(\omega_{-1}, \omega_{-2}, \ldots, \omega_{-k}) > 0\}.$$

Therefore, in the n-gram language model, we need estimates of $\mathsf{Pr}[v \mid \boldsymbol{h}_k]$, whenever $C_d(\boldsymbol{h}_k) > 0$, and for all [4] $k = 0, 1, 2, \ldots, n-1$. By minimizing $\mathscr{LP}_d$, we obtain maximum likelihood estimates of such probabilities, given by (Jelinek, 1997)

$$\mathsf{Pr}_{\mathsf{ML}}[v \mid \boldsymbol{h}_k] = \frac{C_d(v, \boldsymbol{h}_k)}{C_d(\boldsymbol{h}_k)} \quad \text{for all } v \in \upsilon. \tag{3}$$

The resulting (based on Eq. (3)) minus log-probability on the development set, $\mathscr{LP}_d$, equals its *entropy* (Shannon, 1951), which we denote by $\mathscr{H}_d$. Notice that, in contrast to the decision tree based language models, discussed in Section 3, minimization of $\mathscr{LP}_d$ (or, equivalently, of $\mathscr{H}_d$, given Eq. (3)) is not explicitly used to obtain the n-gram language model equivalence classification $\boldsymbol{\Phi}$. Of course, $\mathscr{H}_d$ is a decreasing function of n (see also Table 1).

Clearly, smoothing of the probability mass functions (3) is necessary, in order to provide the desired language model generalization to unseen data. Three classes of such smoothing algorithms are discussed next. Experimental results on the Brown corpus are presented in Section 2.4.

2.1. Laws of succession

Laws of succession constitute a group of smoothing techniques, where probability mass functions $\mathsf{Pr}[v \mid \boldsymbol{h}_k]$, with $0 \leqslant k < n$ and $C_d(\boldsymbol{h}_k) > 0$, are estimated only by means of counts $C_d(v, \boldsymbol{h}_k)$, for all $v \in \upsilon$, as well as, by means of certain parameters that are either a priori chosen, or optimized by taking into account held-out set event counts. Therefore, no information is used from a coarser partitioning of the development set history space.

A variety of laws of succession exist in the literature (Bell et al., 1990; Good, 1953; Ney et al., 1995; Ristad, 1995). We have carried out a detailed comparison of a number of them, in the case study of letter n-gram language modeling on the Brown corpus (Potamianos and Jelinek, 1997). The laws of succession listed below consistently outperformed the rest. We denote by $\boldsymbol{h}$ any history $\boldsymbol{h}_k$ with $C_d(\boldsymbol{h}_k) > 0$, where $0 \leqslant k < n$, and by $q(\boldsymbol{h}) = |\{v \in \upsilon: C_d(v, \boldsymbol{h}) > 0\}|$.

- *Sharpened uniform cardinality prior* based law (Ristad, 1995), given by

[3] This report can be downloaded from http://www.research.att.com/~makis/CLSP.97.13.ps.

[4] The case $k = 0$ corresponds to the uni-gram. In such a case, $C_d(\upsilon, \boldsymbol{h}_0) = C_d(\upsilon)$.

$$\mathsf{Pr}_{\mathsf{LOS.1}}[v \mid \boldsymbol{h}] = \begin{cases} \frac{C_d(v,\boldsymbol{h})}{C_d(\boldsymbol{h})} & \text{if } q(\boldsymbol{h}) = \mid \upsilon \mid; \text{ otherwise:} \\ \frac{C_d(v,\boldsymbol{h})}{C_d(\boldsymbol{h})} \frac{C_d(\boldsymbol{h})[C_d(\boldsymbol{h})+1]+q(\boldsymbol{h})[1-q(\boldsymbol{h})]}{[C_d^2(\boldsymbol{h})+C_d(\boldsymbol{h})+2q(\boldsymbol{h})]} & \text{if } C_d(v,\boldsymbol{h}) > 0, \\ \frac{q(\boldsymbol{h})[q(\boldsymbol{h})+1]}{[\mid\upsilon\mid-q(\boldsymbol{h})][C_d^2(\boldsymbol{h})+C_d(\boldsymbol{h})+2q(\boldsymbol{h})]} & \text{if } C_d(v,\boldsymbol{h}) = 0. \end{cases} \tag{4a}$$

- *Absolute discounting by half* law (Bell et al., 1990; Ney et al., 1995), given by

$$\mathsf{Pr}_{\mathsf{LOS.2}}[v \mid \boldsymbol{h}] = \begin{cases} \frac{C_d(v,\boldsymbol{h})}{C_d(\boldsymbol{h})} & \text{if } q(\boldsymbol{h}) = \mid \upsilon \mid; \quad \text{otherwise:} \\ \frac{C_d(v,\boldsymbol{h})-0.5}{C_d(\boldsymbol{h})} & \text{if } C_d(v,\boldsymbol{h}) > 0, \\ \frac{0.5q(\boldsymbol{h})}{[\mid\upsilon\mid-q(\boldsymbol{h})]C_d(\boldsymbol{h})} & \text{if } C_d(v,\boldsymbol{h}) = 0. \end{cases} \tag{4b}$$

- *Absolute discounting* law (Ney et al., 1995), which is a generalization of Eq. (4b) using a discounting constant $0 < \delta(\boldsymbol{h}) < 1$,

$$\mathsf{Pr}_{\mathsf{LOS.3}}[v \mid \boldsymbol{h}] = \begin{cases} \frac{C_d(v,\boldsymbol{h})}{C_d(\boldsymbol{h})} & \text{if } q(\boldsymbol{h}) = \mid \upsilon \mid; \text{ otherwise:} \\ \frac{C_d(v,\boldsymbol{h})-\delta(\boldsymbol{h})}{C_d(\boldsymbol{h})} & \text{if } C_d(v,\boldsymbol{h}) > 0, \\ \frac{\delta(\boldsymbol{h})q(\boldsymbol{h})}{[\mid\upsilon\mid-q(\boldsymbol{h})]C_d(\boldsymbol{h})} & \text{if } C_d(v,\boldsymbol{h}) = 0. \end{cases} \tag{4c}$$

In Eq. (4c), the smoothing coefficients $\delta(\boldsymbol{h})$ have to be obtained for the histories $\boldsymbol{h}$ of interest. It is reasonable to choose these coefficients to minimize the minus log-probability on the held-out set, $\mathscr{LP}_c$ (see Eq. (2)), hoping that this will provide good generalization of the resulting language model to the unseen test set (i.e., result in a small $\mathscr{LP}_t$). However, we need to guard against over-fitting of these parameters to the held-out data set. We can meet both goals by *bucketing* the histories $\boldsymbol{h}$ of interest in terms of a chosen feature, for example, the development set history count $C_d(\boldsymbol{h})$, as suggested by Bahl et al. (1983), while simultaneously assuring that enough held-out set events are contained within the constructed history *buckets*. All histories within a bucket are assigned a single value of the smoothing constant. Features alternative to $C_d(\boldsymbol{h})$ can also be used to construct history buckets. Indeed, the use of $C_d(\boldsymbol{h})/q(\boldsymbol{h})$ is advocated by Chen and Goodman (1996), whereas, the two-dimensional feature $(C_d(\boldsymbol{h}), q(\boldsymbol{h}))$ is used by Ristad and Thomas (1997b). In the following, we briefly discuss our bucketing algorithm in terms of feature $C_d(\boldsymbol{h})$ (Potamianos and Jelinek, 1997), and its application to estimating $\delta(\boldsymbol{h})$ in Eq. (4c). Extensions to other history bucketing features can be made in a straightforward manner.

Let $\mathscr{C}$ denote the set of histories $\boldsymbol{h}$ of interest. For example (see Eq. (4c)), $\mathscr{C} = \{\boldsymbol{h}_k : C_d(\boldsymbol{h}_k) > 0, \text{ and } q(\boldsymbol{h}_k) < \mid \upsilon \mid\}$. We now seek integer constants $B_0, B_1, \ldots, B_L$, that partition the set $\mathscr{C}$ into L sets (buckets)

$$S_j = \{\boldsymbol{h} \in \mathscr{C}: \; B_{j-1} \leqslant C_d(\boldsymbol{h}) < B_j\} \quad \text{for } j = 1, 2, \ldots, L\,. \tag{5}$$

Such constants are chosen to satisfy

$$\min_{\boldsymbol{h}\in\mathscr{C}}\{C_d(\boldsymbol{h})\} = B_0 < B_1 < B_2 < \cdots < B_{L-1} < B_L = \max_{\boldsymbol{h}\in\mathscr{C}}\{C_d(\boldsymbol{h})\} + 1\,, \tag{6a}$$

$$\sum_{\boldsymbol{h}\in S_j} C_c(\boldsymbol{h}) \geqslant C_{\min}, \tag{6b}$$

and, approximately (subject to Eqs. (6a) and (6b)), [5]

[5] We denote $\lceil x \rceil = \min\{i : \; i \text{ is integer, and } i \geqslant x\}$.

$$B_j \simeq \lceil bB_{j-1} \rceil, \tag{6c}$$

for $j = 1, 2, \ldots, L$. In Eqs. (6a)–(6c), $b > 1$ is a real constant that controls the "width" of buckets (5), whereas $C_{\min}$ denotes the minimum required number of held-out set events in bucket S_j to ensure reliable estimation of tied $\delta_j = \delta(\boldsymbol{h})$, for $\boldsymbol{h} \in S_j$, $j = 1, 2, \ldots, L$. [6] The tied smoothing coefficients are then determined by minimizing $\mathscr{L}\mathscr{P}_c$ (equivalently, by maximizing $-\mathscr{L}\mathscr{P}_c$), as (see Eqs. (2) and (4c))

$$\hat{\delta}_j = \arg\max_{0<\delta_j<1} \left\{ \sum_{\boldsymbol{h}\in S_j} \sum_{v\in\nu: C_d(v,\boldsymbol{h})>0} C_c(v,\boldsymbol{h}) \log_2 \frac{C_d(v,\boldsymbol{h}) - \delta_j}{C_d(\boldsymbol{h})} + \sum_{\boldsymbol{h}\in S_j} \sum_{v\in\nu: C_d(v,\boldsymbol{h})=0} C_c(v,\boldsymbol{h}) \log_2 \frac{\delta_j q(\boldsymbol{h})}{[\,|\,\nu\,| - q(\boldsymbol{h})] C_d(\boldsymbol{h})} \right\}, \tag{7a}$$

for $j = 1, 2, \ldots, L$. Optimization (7a) can be simply carried out by means of the *Newton–Raphson* algorithm (Press et al., 1988), or the simpler, but slower, *bisection* method (Press et al., 1988), for solving

$$\sum_{\boldsymbol{h}\in S_j} \sum_{v\in\nu: C_d(v,\boldsymbol{h})>0} C_c(v,\boldsymbol{h}) \, [\delta_j - C_d(v,\boldsymbol{h})]^{-1} + \sum_{\boldsymbol{h}\in S_j} \sum_{v\in\nu: C_d(v,\boldsymbol{h})=0} C_c(v,\boldsymbol{h}) \, \delta_j^{-1} = 0, \tag{7b}$$

in the compact set $[\epsilon, 1-\epsilon]$, where ϵ is a small positive number, such as $\epsilon = 10^{-5}$. Notice that there exists a unique solution to Eq. (7b), because its left-hand side is the first derivative of the argument of Eq. (7a), which is a concave function with respect to δ_j.

2.2. The back-off approach

The main mechanism of the back-off smoothing algorithm (Katz, 1987; Ney et al., 1995) is the discounting of the maximum likelihood estimates (3) of seen event probabilities $\mathsf{Pr}_{\mathsf{ML}}[v \mid \boldsymbol{h}_k]$ by means of any law of succession and the assignment of the remaining probability mass to the unseen events (such that $C_d(v, \boldsymbol{h}_k) = 0$ and $C_d(\boldsymbol{h}_k) > 0$), not in a uniform way (as is the case with any law of succession), but according to the back-off smoothed probability mass function of the immediate ancestor history $\boldsymbol{h}_{k-1}$ of the history $\boldsymbol{h}_k$ under consideration.

The back-off smoothed estimates of $\mathsf{Pr}[v \mid \boldsymbol{h}_k]$, for all $v \in \nu$, whenever $C_d(\boldsymbol{h}_k) > 0$, with $0 \leqslant k \leqslant n-1$, are defined by the initialization [7]

$$\mathsf{Pr}_{\mathsf{BOF}}[v \mid \boldsymbol{h}_0] = \mathsf{Pr}_{\mathsf{LOS}}[v \mid \boldsymbol{h}_0], \tag{8a}$$

and the top–down recursion, for $i = 1, 2, \ldots, k$,

$$\mathsf{Pr}_{\mathsf{BOF}}[v \mid \boldsymbol{h}_i] = \begin{cases} \mathsf{Pr}_{\mathsf{LOS}}[v \mid \boldsymbol{h}_i] & \text{if } q(\boldsymbol{h}_i) = |\,\nu\,|; \text{ otherwise:} \\ \mathsf{Pr}_{\mathsf{LOS}}[v \mid \boldsymbol{h}_i] & \text{if } C_d(v, \boldsymbol{h}_i) > 0, \\ \beta(\boldsymbol{h}_i) \mathsf{Pr}_{\mathsf{BOF}}[v \mid \boldsymbol{h}_{i-1}] & \text{if } C_d(v, \boldsymbol{h}_i) = 0, \end{cases} \tag{8b}$$

where, if $q(\boldsymbol{h}_i) < |\,\nu\,|$,

$$\beta(\boldsymbol{h}_i) = \frac{1 - \sum_{v\in\nu:\ C_d(v,\boldsymbol{h}_i)>0} \mathsf{Pr}_{\mathsf{LOS}}[v \mid \boldsymbol{h}_i]}{\sum_{v\in\nu:\ C_d(v,\boldsymbol{h}_i)=0} \mathsf{Pr}_{\mathsf{BOF}}[v \mid \boldsymbol{h}_{i-1}]}, \tag{8c}$$

[6] Throughout our language modeling experiments on the Brown corpus, and whenever smoothing algorithms require bucketing for optimally determining smoothing coefficients, we have used $C_{\min} = \min\{\lceil |\,\nu\,|/4 \rceil, \sum_{\boldsymbol{h}\in\mathscr{C}} C_c(\boldsymbol{h})\}$ and $b = 1.2$.

[7] In Eqs. (8a)–(8c), we denote by $\boldsymbol{h}_{i-1}$ the immediate ancestor of history $\boldsymbol{h}_i$; for example, if $\boldsymbol{h}_i = (\omega_{-1}, \omega_{-2}, \ldots, \omega_{-i})$, then $\boldsymbol{h}_{i-1} = (\omega_{-1}, \omega_{-2}, \ldots, \omega_{-i+1})$. Notice that $\boldsymbol{h}_0$ corresponds to the uni-gram.

which guarantees that Eq. (8b) is a valid probability mass function. In Eqs. (8a)–(8c), any law of succession (4a)–(4c) could be used. We denote the resulting back-off probability mass function accordingly; for example, by $\mathsf{Pr}_{\mathsf{BOF.1}}[v \mid \boldsymbol{h}_k]$, if $\mathsf{Pr}_{\mathsf{LOS.1}}[v \mid \boldsymbol{h}_k]$ is employed. Clearly, when using back-off smoothing in combination with Eq. (4c), we need to estimate the smoothing parameters $\delta(\boldsymbol{h}_i)$. It is not hard to see that Eq. (7b) is applicable here as well. However, Eqs. (6a)–(6c) and (7b) should be consecutively applied to seen histories of depths $0, 1, \ldots, n-1$, in a top–down fashion (Potamianos and Jelinek, 1997).

2.3. Deleted interpolation

Let us again consider any history $\boldsymbol{h}_k$, such that $C_d(\boldsymbol{h}_k) > 0$ and $0 \leqslant k \leqslant n-1$. We are interested in determining $\mathsf{Pr}[v \mid \boldsymbol{h}_k]$, for all $v \in \upsilon$. For such a history, we consider the series of its ancestors $\boldsymbol{h}_i$, and maximum likelihood estimates $\mathsf{Pr}_{\mathsf{ML}}[v \mid \boldsymbol{h}_i]$ (see Eq. (3)), for $i = k-1, k-2, \ldots, 1, 0$. In addition, we consider the "null" history $\boldsymbol{h}_{-1} = \emptyset$, with $\mathsf{Pr}_{\mathsf{ML}}[v \mid \boldsymbol{h}_{-1}] = 1/|\upsilon|$. The deleted interpolation algorithm (Bahl et al., 1983; Jelinek and Mercer, 1980), estimates the desired $\mathsf{Pr}[v \mid \boldsymbol{h}_k]$, $v \in \upsilon$, as

$$\mathsf{Pr}_{\mathsf{DI}}[v \mid \boldsymbol{h}_k] = \sum_{i=-1}^{k} \lambda^{(i)}(\boldsymbol{h}_k)\, \mathsf{Pr}_{\mathsf{ML}}[v \mid \boldsymbol{h}_i], \tag{9a}$$

where $\lambda^{(i)}(\boldsymbol{h}_k)$ are smoothing coefficients that satisfy

$$0 < \lambda^{(i)}(\boldsymbol{h}_k) < 1, \quad \text{for } i = -1, 0, 1, 2, \ldots, k, \quad \text{and } \sum_{i=-1}^{k} \lambda^{(i)}(\boldsymbol{h}_k) = 1\,. \tag{9b}$$

The smoothing coefficients should be optimally chosen to minimize the minus log-probability on the held-out data set, $\mathscr{LP}_c$ (see Eq. (2)). In this paper, we consider two algorithms for estimating these coefficients, namely the *top-down* and the *bottom-up* deleted interpolation algorithms. We denote the resulting probability mass functions by $\mathsf{Pr}_{\mathsf{DI.TD}}[v \mid \boldsymbol{h}_k]$ and $\mathsf{Pr}_{\mathsf{DI.BU}}[v \mid \boldsymbol{h}_k]$, respectively.

2.3.1. Top-down deleted interpolation

The top–down deleted interpolation algorithm sequentially derives $\mathsf{Pr}_{\mathsf{DI.TD}}[v \mid \boldsymbol{h}_0]$, $\mathsf{Pr}_{\mathsf{DI.TD}}[v \mid \boldsymbol{h}_1]$, $\ldots, \mathsf{Pr}_{\mathsf{DI.TD}}[v \mid \boldsymbol{h}_k]$, for all histories $\boldsymbol{h}_k$ of interest, as illustrated in Fig. 1(a). In detail, let us define sets

$$\mathscr{C}_i = \{\boldsymbol{h}_i \in \upsilon^i\colon C_d(\boldsymbol{h}_i) > 0\} \quad \text{for } i = 0, 1, \ldots, n-1 \tag{10a}$$

(clearly, $\mathscr{C}_0$ is a singleton), and apply Eqs. (6a)–(6c) to obtain history buckets

$$S_{i,j} = \{\boldsymbol{h}_i \in \mathscr{C}_i\colon B_{i,j-1} \leqslant C_d(\boldsymbol{h}_i) < B_{i,j}\} \quad \text{for } j = 1, 2, \ldots, L_i\,, \;\; i = 0, 1, \ldots, n-1. \tag{10b}$$

Then, the top-down deleted interpolation algorithm consists of the initialization

$$\mathsf{Pr}_{\mathsf{DI.TD}}[v \mid \boldsymbol{h}_{-1}] = \mathsf{Pr}_{\mathsf{ML}}[v \mid \boldsymbol{h}_{-1}] = \frac{1}{|\upsilon|}, \tag{11a}$$

and the following two steps, iterated over all buckets $S_{i,j}$, $j = 1, 2, \ldots, L_i$, and over all levels $i = 0, 1, \ldots, n-1$: the optimization step (see also Eq. (2))

$$\hat{\lambda}_{i,j} = \arg\max_{0<\lambda<1} \left\{ \sum_{\boldsymbol{h}_i \in S_{i,j}} \sum_{v \in \upsilon} C_c(v, \boldsymbol{h}_i) \log_2[\lambda \mathsf{Pr}_{\mathsf{DI.TD}}[v \mid \boldsymbol{h}_{i-1}] + (1-\lambda)\mathsf{Pr}_{\mathsf{ML}}[v \mid \boldsymbol{h}_i]] \right\}, \tag{11b}$$

and the substitution step

$$\mathsf{Pr}_{\mathsf{DI.TD}}[v \mid \boldsymbol{h}_i] = \hat{\lambda}_{i,j} \mathsf{Pr}_{\mathsf{DI.TD}}[v \mid \boldsymbol{h}_{i-1}] + (1 - \hat{\lambda}_{i,j}) \mathsf{Pr}_{\mathsf{ML}}[v \mid \boldsymbol{h}_i], \tag{11c}$$

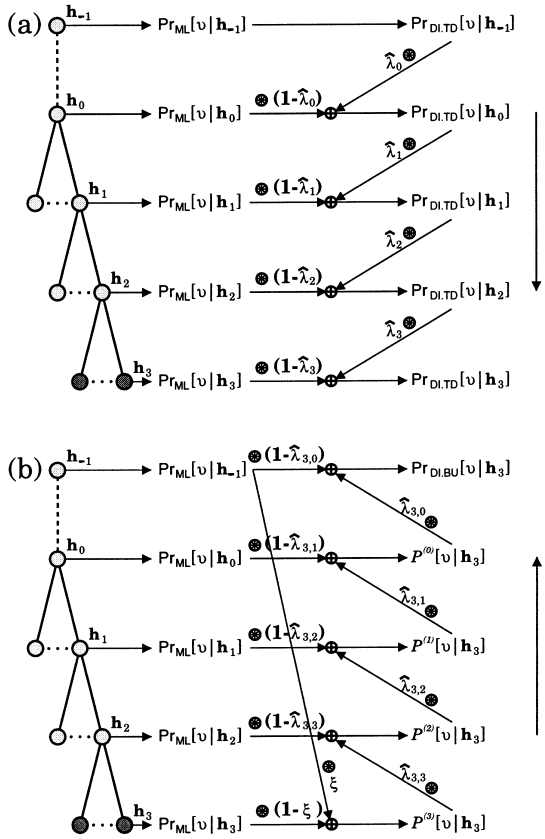

Fig. 1. Demonstration of the two deleted interpolation algorithms, discussed in Section 2.3, in the case $k = 3$: (a) top–down algorithm; (b) bottom–up algorithm.

where, in Eq. (11c), $\boldsymbol{h}_i \in S_{i,j}$. Optimization (11b) can be easily carried out, by means of the Newton–Raphson algorithm for solving

$$\sum_{\boldsymbol{h}_i \in S_{i,j}} \sum_{v \in \mathcal{V}} C_c(v, \boldsymbol{h}_i) \frac{\mathsf{Pr}_{\mathsf{DI.TD}}[v \mid \boldsymbol{h}_{i-1}] - \mathsf{Pr}_{\mathsf{ML}}[v \mid \boldsymbol{h}_i]}{\lambda[\mathsf{Pr}_{\mathsf{DI.TD}}[v \mid \boldsymbol{h}_{i-1}] - \mathsf{Pr}_{\mathsf{ML}}[v \mid \boldsymbol{h}_i]] + \mathsf{Pr}_{\mathsf{ML}}[v \mid \boldsymbol{h}_i]} = 0, \tag{11d}$$

in the compact set $[\epsilon, 1-\epsilon]$ (e.g., $\epsilon = 10^{-5}$). Alternatively, the bisection method can be used to solve Eq. (11d) (Bahl et al., 1991). Similar to solving Eq. (7b), solving Eq. (11d) is tractable, because its left-hand side is the first derivative of the argument of Eq. (11b), which is a concave function with respect to λ. The final smoothing coefficients of Eqs. (9a) and (9b) are given by

$$\lambda^{(i)}(\boldsymbol{h}_k) = \begin{cases} \prod_{k'=0}^{k} \hat{\lambda}_{k',j_{k'}} & \text{if } i = -1, \\ (1 - \hat{\lambda}_{i,j_i}) \prod_{k'=i+1}^{k} \hat{\lambda}_{k',j_{k'}} & \text{if } 0 \leqslant i \leqslant k, \end{cases} \tag{11e}$$

where $j_{k'} = j$, whenever $\boldsymbol{h}_{k'} \in S_{k',j}$, where $0 \leqslant k' \leqslant k$ (Potamianos and Jelinek, 1997).

2.3.2. Bottom–up deleted interpolation

The bottom–up deleted interpolation algorithm constructs a series of estimates of probability mass function $\mathsf{Pr}_{\mathsf{DI.BU}}[v \mid \boldsymbol{h}_k]$, $v \in \upsilon$, for all histories $\boldsymbol{h}_k$ that satisfy $C_d(\boldsymbol{h}_k) > 0$ and $0 \leqslant k \leqslant n-1$. Let us denote such estimates by $P^{(i)}[v \mid \boldsymbol{h}_k]$, $v \in \upsilon$, for $i = k, k-1, \ldots, 0, -1$. The initial estimate $P^{(k)}[v \mid \boldsymbol{h}_k]$ of $\mathsf{Pr}_{\mathsf{DI.BU}}[v \mid \boldsymbol{h}_k]$ is a slightly smoothed version of the maximum likelihood estimate $\mathsf{Pr}_{\mathsf{ML}}[v \mid \boldsymbol{h}_k]$ at level k (see Eq. (12a) below), whereas, the final estimate $P^{(-1)}[v \mid \boldsymbol{h}_k]$ is the desired smoothed probability mass function $\mathsf{Pr}_{\mathsf{DI.BU}}[v \mid \boldsymbol{h}_k]$ (see also Fig. 1(b)).

In detail, suppose that, for some $0 \leqslant k \leqslant n-1$, we wish to estimate $\mathsf{Pr}_{\mathsf{DI.BU}}[v \mid \boldsymbol{h}_k]$, whenever $C_d(\boldsymbol{h}_k) > 0$. First, we bucket the histories in set $\mathscr{C}_k$ (see Eq. (10a)) into sets (buckets) $S_{k,j}$, $j = 1, 2, \ldots, L_k$ (see Eq. (10b)), by using Eqs. (6a)–(6c). Then, the bottom–up deleted interpolation algorithm consists of the initialization step

$$P^{(k)}[v \mid \boldsymbol{h}_k] = (1-\xi)\mathsf{Pr}_{\mathsf{ML}}[v \mid \boldsymbol{h}_k] + \xi\, \mathsf{Pr}_{\mathsf{ML}}[v \mid \boldsymbol{h}_{-1}], \tag{12a}$$

where $0 < \xi < 1$ is a sufficiently small positive parameter, chosen in an ad-hoc manner (e.g., $\xi = 10^{-5}$), and the following two steps, iterated over all buckets $S_{k,j}$, $j = 1, 2, \ldots, L_k$, and over all $i = k, k-1, \ldots, 1, 0$: the optimization step (see also Eq. (2))

$$\hat{\lambda}_{k,i,j} = \arg\max_{0<\lambda<1} \left\{ \sum_{\boldsymbol{h}_k \in S_{k,j}} \sum_{v \in \upsilon} C_c(v, \boldsymbol{h}_k) \log_2 [\lambda P^{(i)}[v \mid \boldsymbol{h}_k] + (1-\lambda)\mathsf{Pr}_{\mathsf{ML}}[v \mid \boldsymbol{h}_{i-1}]] \right\}, \tag{12b}$$

performed by means of the Newton–Raphson algorithm, or the bisection method (similarly to the solution of Eqs. (7b) and (11d)), as a solution of

$$\sum_{\boldsymbol{h}_k \in S_{k,j}} \sum_{v \in \upsilon} C_c(v, \boldsymbol{h}_k) \frac{P^{(i)}[v \mid \boldsymbol{h}_k] - \mathsf{Pr}_{\mathsf{ML}}[v \mid \boldsymbol{h}_{i-1}]}{\lambda[P^{(i)}[v \mid \boldsymbol{h}_k] - \mathsf{Pr}_{\mathsf{ML}}[v \mid \boldsymbol{h}_{i-1}]] + \mathsf{Pr}_{\mathsf{ML}}[v \mid \boldsymbol{h}_{i-1}]} = 0,$$

in the compact set $[\epsilon, 1-\epsilon]$ (e.g., $\epsilon = 10^{-5}$), and the substitution step

$$P^{(i-1)}[v \mid \boldsymbol{h}_k] = \hat{\lambda}_{k,i,j} P^{(i)}[v \mid \boldsymbol{h}_k] + (1-\hat{\lambda}_{k,i,j})\mathsf{Pr}_{\mathsf{ML}}[v \mid \boldsymbol{h}_{i-1}], \tag{12c}$$

where, in Eq. (12c), $\boldsymbol{h}_k \in S_{k,j}$. Finally,

$$\mathsf{Pr}_{\mathsf{DI.BU}}[v \mid \boldsymbol{h}_k] = P^{(-1)}[v \mid \boldsymbol{h}_k]\ , \quad \text{for all } \ v \in \upsilon. \tag{12d}$$

The final smoothing coefficients of Eqs. (9a) and (9b) are given by

$$\lambda^{(i)}(\boldsymbol{h}_k) = \begin{cases} 1 - \hat{\lambda}_{k,0,j_k} + \prod_{k'=0}^{k+1} \hat{\lambda}_{k,k',j_k} & \text{if } i = -1, \\ (1 - \hat{\lambda}_{k,i+1,j_k}) \prod_{k'=0}^{i} \hat{\lambda}_{k,k',j_k} & \text{if } 0 \leqslant i \leqslant k, \end{cases} \tag{12e}$$

where $j_k = j$, whenever $\boldsymbol{h}_k \in S_{k,j}$, and $\hat{\lambda}_{k,k+1,j_k} = \xi$. Notice that Eqs. (12a)–(12e) have to be calculated for all $k = 0, 1, \ldots, n-1$ (Potamianos and Jelinek, 1997).

Obviously, the top–down deleted interpolation algorithm is computationally less intensive than the bottom–up one. This is due to the fact that the former naturally determines $\mathsf{Pr}_{\mathsf{DI.TD}}[v \mid \boldsymbol{h}_{i-1}]$, before determining $\mathsf{Pr}_{\mathsf{DI.TD}}[v \mid \boldsymbol{h}_i]$. However, the latter algorithm gives rise to superior language models, possibly because the history bucketing, as well as, successive optimizations (12b) always occur at the finest level of interest k.

Notice that both top–down and bottom–up deleted interpolation algorithms avoid the simultaneous optimization of all smoothing coefficients, and, therefore, they are sub-optimal. Alternatively, one can choose to simultaneously estimate all $k+2$ such coefficients in Eq. (9a), by maximizing the held-out data log-probability, $-\mathscr{L}\mathscr{P}_c$, over the simplex (9b). This is a concave function with respect to the vector of smoothing

coefficients (Bahl et al., 1991), therefore, a unique solution to the problem can be found by means of a *conjugate gradient* method (Press et al., 1988). The desired smoothing coefficients can be tied as in the bottom-up deleted interpolation algorithm. However, our preliminary experiments indicate that this approach results in insignificant improvement (over the use of Eqs. (12a)–(12e)) in both held-out and test set log-probabilities, and at a significantly higher computational cost, as k becomes larger. Alternatively, one can use the *expectation-maximization* algorithm to simultaneously estimate the smoothing coefficients in Eqs. (9a) and (9b) (Jelinek and Mercer, 1980; Ristad and Thomas, 1997b). However, this method has been reported to be significantly more computationally expensive that optimization (11b), even in the simple case $k = 0$, considered by Bahl et al. (1991). In addition, Ristad and Thomas (1997b) compare the use of the expectation-maximization algorithm to our bottom-up deleted interpolation technique for n-gram language model smoothing, and they report no significant performance difference between the two methods.

2.4. Experimental results

We now compare the performance of the three laws of succession (4a)–(4c), their three back-off variants (8a)–(8c), and the two deleted interpolation smoothing algorithms (11a)–(11e), (12a)–(12e), in the case of the n-gram letter language model on the Brown corpus, for $n = 1, 2, \ldots, 10$. In Table 1, we depict the test set minus log-probability, $\mathscr{LP}_t$, of the resulting language models, as well as, the development set entropy, $\mathscr{H}_d$.

As expected, the laws of succession perform the worst. The resulting n-gram models are over-trained for $n > 6$. The absolute discounting law (LOS.3) is slightly better than the other laws of succession, since it allows estimation of the smoothing coefficients $\delta(\boldsymbol{h})$ on a held-out set. The back-off smoothing technique performs significantly better than the laws of succession. Back-off smoothing based on the absolute discounting law (BOF.3) is slightly better than the rest (BOF.1 and BOF.2). However, the resulting n-gram models are over-trained for $n > 7$. On the other hand, both deleted interpolation algorithms exhibit robustness to sparseness of the training data, with the bottom–up algorithm achieving smaller minus log-probability on the test set.

Our best n-gram language model achieves a minus log-probability of 1.796 bits per character on the Brown corpus test set, and it is obtained by smoothing the 10-gram language model by means of the bottom–up deleted interpolation algorithm. This result is better than the 1.91 bits per character achieved by the context model of Ristad and Thomas (1995), but worse that the 1.75 bits per character reported by Brown et al. (1992a). There, a sophisticated tri-gram word language model with dedicated spelling, case, and spacing sub-models is employed, and, moreover, 600 times more training data, obtained from various corpora,

Table 1
Minus log-probability on our Brown corpus test subset, $\mathscr{LP}_t$, of the n-gram language model (smoothed by the algorithms discussed in Section 2, for $n = 1, 2, \ldots, 10$)

n	$\mathscr{H}_d$	LOS.1	LOS.2	LOS.3	BOF.1	BOF.2	BOF.3	DI.TD	DI.BU
1	4.378	4.380	4.380	4.380	4.380	4.380	4.380	4.380	4.380
2	3.469	3.471	3.471	3.470	3.470	3.471	3.470	3.470	3.470
3	2.847	2.851	2.852	2.851	2.851	2.851	2.850	2.851	2.850
4	2.288	2.333	2.334	2.331	2.327	2.327	2.324	2.328	2.326
5	1.883	2.038	2.040	2.034	2.007	2.008	2.002	2.016	2.007
6	1.609	**1.979**	**1.983**	**1.975**	1.884	1.886	1.876	1.894	1.878
7	1.375	2.071	2.074	2.065	**1.864**	**1.867**	**1.854**	1.853	1.831
8	1.135	2.233	2.236	2.227	1.888	1.891	1.877	1.837	1.811
9	0.899	2.401	2.403	2.394	1.927	1.930	1.915	1.828	1.801
10	**0.687**	2.554	2.556	2.539	1.971	1.974	1.948	**1.824**	**1.796**

The development set entropy, $\mathscr{H}_d$, is also listed. All results are measured in bits per character.

is used. Finally, the hierarchical non-emitting Markov language model, introduced by Ristad and Thomas (1997a,b), achieves 1.73 bits per character on the same task. This model does not satisfy Eq. (1), and it is shown to be strictly more powerful than the n-gram language model. We believe that part of its superior performance is due to the use of N-fold cross validation, re-estimation of Eq. (3) on basis of the entire training set, and employment of a two-dimensional feature history bucketing scheme, when estimating necessary smoothing coefficients (Ristad and Thomas, 1997b).

2.5. Remarks on smoothing algorithms

In Section 2, we presented a variety of smoothing algorithms for language models. Among them, we concluded that the bottom–up deleted interpolation is the most suitable for our n-gram letter language models on the Brown corpus, being very robust to data sparseness. Both back-off smoothing algorithms and laws of succession fail to exhibit this robustness, with the back-off smoothing algorithm being significantly better than its corresponding law of succession, especially for sparse data. Interestingly, the back-off scheme based on absolute discounting compares favorably to the bottom–up deleted interpolation algorithm for values of $n \leqslant 6$, thus being a viable alternative to it. However, when the training data is really sparse ($n \geqslant 7$), one has to resort to the safety of the bottom–up deleted interpolation algorithm, which significantly outperforms the back-off smoothing algorithm.

These conclusions are in line with experimental results reported in Chen and Goodman (1996). There, and for three word language modeling tasks, the back-off smoothed word n-gram language models are consistently outperformed in sparse training data domains by the deleted interpolation smoothed ones (word tri-gram language models, trained on "insufficient" data). This is typically reversed in the case of bi-gram word language models, where, clearly, the training data domain is not as sparse. A possible cause for the inferior performance of back-off smoothing in sparse data domains is the fact that Eqs. (4c) and (8b) do not discount the development set counts $C_d(v, \boldsymbol{h})$ enough (recall that $0 < \delta(\boldsymbol{h}) < 1$), thus, favoring higher order history events over lower order ones. A remedy to this problem could be to discount $C_d(v, \boldsymbol{h})$ by $\delta(v, \boldsymbol{h})$, instead of $\delta(\boldsymbol{h})$, in Eq. (4c), where $0 < \delta(v, \boldsymbol{h}) < C_d(v, \boldsymbol{h})$. Such discounting schemes have not been considered in this paper. It is worth mentioning that our history bucketing scheme, as implemented in this work, has improved the performance of back-off smoothing based on the absolute discounting law, over the implementation reported by Ney et al. (1995). There, for each history level k, all discounting coefficients are tied to a single value; i.e., $\delta_k = \delta(\boldsymbol{h}_k)$, for all $\boldsymbol{h}_k$ with $C_d(\boldsymbol{h}_k) > 0$ and $0 \leqslant k \leqslant n-1$ (see also Eq. (4c)). Similarly to our experimental results, Ney et al. (1995) report that the absolute discounting law is the most appropriate for back-off smoothing.

So far, our exposition has been in reference to the n-gram language model. However, the above smoothing algorithms are readily applicable to more general language models. Indeed, in Section 3.4, we discuss bottom-up deleted interpolation and back-off smoothing for decision tree language models.

3. Decision tree based language modeling

Decision trees constitute popular classifiers (Breiman et al., 1984). Since the language modeling problem (1) is a classification problem, it is natural to investigate decision tree based language models. Such models have been developed before (Bahl et al., 1989), however, our work is novel in the following aspects: (a) we apply K-means clustering for decision tree *question* design (Chou, 1991); (b) we compare this algorithm to one based on a *binary encoding* of the vocabulary (Jelinek, 1997); (c) we propose a robust decision tree smoothing algorithm; and (d) we provide a fair comparison between decision tree and n-gram language models. Experimental results on the Brown corpus are presented in Section 3.5.

3.1. Decision tree notation and design issues

A *tree* $\boldsymbol{T}$ consists of a finite collection of *nodes* $\boldsymbol{T} = \{\boldsymbol{t}_0, \boldsymbol{t}_1, \ldots\}$, where $\boldsymbol{t}_0$ denotes the tree *root*. Every node has a unique *parent* node, denoted by $\mathrm{PAR}(\boldsymbol{t}) \in \boldsymbol{T}$ (except for $\mathrm{PAR}(\boldsymbol{t}_0) = \mathrm{NULL}$), and $K(\boldsymbol{t}) \geqslant 0$ *children* nodes, denoted by $\mathrm{KID}_i(\boldsymbol{t}) \in \boldsymbol{T}$, for $i = 1, 2, \ldots, K(\boldsymbol{t})$. For every node $\boldsymbol{t} \in \boldsymbol{T}$, we recursively define its *depth*, as

$$\mathrm{DEPTH}(\boldsymbol{t}) = \begin{cases} 0 & \text{if } \boldsymbol{t} = \boldsymbol{t}_0, \\ \mathrm{DEPTH}(\mathrm{PAR}(\boldsymbol{t})) + 1 & \text{otherwise,} \end{cases} \tag{13a}$$

and the set of its *ancestors* $\underline{\mathrm{ANC}}(\boldsymbol{t}) = \{\mathrm{ANC}_i(\boldsymbol{t}):\ i = -1, 0, 1, \ldots, \mathrm{DEPTH}(\boldsymbol{t})\}$, as

$$\mathrm{ANC}_i(\boldsymbol{t}) = \begin{cases} \boldsymbol{t} & \text{if } i = \mathrm{DEPTH}(\boldsymbol{t}); \text{otherwise:} \\ \mathrm{PAR}(\mathrm{ANC}_{i+1}(\boldsymbol{t})) & \text{for } i = \mathrm{DEPTH}(\boldsymbol{t}) - 1, \ldots, 1, 0, -1. \end{cases} \tag{13b}$$

Clearly, $\mathrm{ANC}_0(\boldsymbol{t}) = \boldsymbol{t}_0$, and $\mathrm{ANC}_{-1}(\boldsymbol{t}) = \mathrm{NULL}$. We denote the set of tree *leaves* by $\boldsymbol{L} = \{\boldsymbol{t} \in \boldsymbol{T}: K(\boldsymbol{t}) = 0\} \subseteq \boldsymbol{T}$. Here, we exclusively consider *binary* and *ternary* trees that satisfy $K(\boldsymbol{t}) = 2$ and $K(\boldsymbol{t}) = 3$, for all $\boldsymbol{t} \in \boldsymbol{T} - \boldsymbol{L}$, respectively. In addition, our ternary trees satisfy $\mathrm{KID}_3(\boldsymbol{t}) \in \boldsymbol{L}$, for all $\boldsymbol{t} \in \boldsymbol{T} - \boldsymbol{L}$. Nodes $\mathrm{KID}_3(\boldsymbol{t})$ are called *middle nodes*, and their set is denoted by $\boldsymbol{M} \subset \boldsymbol{L} \subseteq \boldsymbol{T}$ (see also Fig. 2).

A decision tree is a tree $\boldsymbol{T}$ with a question $\mathscr{Q}(\boldsymbol{t})$ assigned at each node $\boldsymbol{t} \in \boldsymbol{T} - \boldsymbol{L}$, and a probability mass function $\mathsf{Pr}[v \mid \boldsymbol{t}]$, assigned at each leaf $\boldsymbol{t} \in \boldsymbol{L}$. Question $\mathscr{Q}(\boldsymbol{t})$ partitions node $\boldsymbol{t}$ into its $K(\boldsymbol{t})$ children nodes, according to a mapping (question) $\mathscr{Q}(\boldsymbol{t}): \boldsymbol{t} \to \{\mathrm{KID}_1(\boldsymbol{t}), \ldots, \mathrm{KID}_{K(\boldsymbol{t})}(\boldsymbol{t})\}$. In the context of language modeling, $v \in \upsilon$, every node $\boldsymbol{t} \in \boldsymbol{T}$ corresponds to a cluster of histories $\boldsymbol{\Phi}_{\boldsymbol{t}}$ (see also Eq. (1)), whereas, every question $\mathscr{Q}(\boldsymbol{t})$ partitions $\boldsymbol{\Phi}_{\boldsymbol{t}}$ into $K(\boldsymbol{t})$ new history clusters $\boldsymbol{\Phi}_{\mathrm{KID}_i(\boldsymbol{t})}$, for $i = 1, 2, \ldots, K(\boldsymbol{t})$. Given the tree $\boldsymbol{T}$, we define (compare to Eq. (3))

$$f(v \mid \boldsymbol{t}) = \mathsf{Pr}_{\mathsf{ML}}[v \mid \boldsymbol{t}] = \frac{C_d(v, \boldsymbol{t})}{C_d(\boldsymbol{t})} \quad \text{and} \quad f(\boldsymbol{t} \mid \boldsymbol{t}') = \frac{C_d(\boldsymbol{t})}{C_d(\boldsymbol{t}')}, \tag{14}$$

for all $v \in \upsilon$, $\boldsymbol{t} \in \boldsymbol{T}$, $\boldsymbol{t}' \in \underline{\mathrm{ANC}}(\boldsymbol{t})$, and $\boldsymbol{t}, \boldsymbol{t}' \neq \mathrm{NULL}$. In Eq. (14), $C_d(v, \boldsymbol{t}) = \sum_{\boldsymbol{h}_{n-1} \in \boldsymbol{\Phi}_{\boldsymbol{t}}} C_d(v, \boldsymbol{h}_{n-1})$, and it denotes the total number of development set occurrences of $v \in \upsilon$, immediately following any of the histories $\boldsymbol{h}_{n-1}$, clustered at node $\boldsymbol{t}$ whereas, $C_d(\boldsymbol{t}) = \sum_{v \in \upsilon} C_d(v, \boldsymbol{t}) = \sum_{\boldsymbol{h}_{n-1} \in \boldsymbol{\Phi}_{\boldsymbol{t}}} C_d(\boldsymbol{h}_{n-1})$.

Similarly to the n-gram language model, and given the equivalence classification $\boldsymbol{\Phi}$ (i.e., the tree $\boldsymbol{T}$), probability mass functions (14), for $\boldsymbol{t} \in \boldsymbol{L}$, minimize the minus log-probability on the development set, $\mathscr{LP}_d$. The resulting $\mathscr{LP}_d$ equals the *tree entropy*, given by (Breiman et al., 1984)

$$\mathscr{H}_d(\boldsymbol{T}) = \sum_{\boldsymbol{t} \in \boldsymbol{L}} f(\boldsymbol{t} \mid \boldsymbol{t}_0) \mathscr{H}_d(\boldsymbol{t}), \tag{15a}$$

where

$$\mathscr{H}_d(\boldsymbol{t}) = -\sum_{v \in \upsilon} f(v \mid \boldsymbol{t}) \log_2 f(v \mid \boldsymbol{t}), \tag{15b}$$

for $\boldsymbol{t} \in \boldsymbol{T}$, denotes the *node entropy* (Shannon, 1951).

In contrast to the n-gram language model, decision tree based language models offer additional freedom in designing the equivalence classification $\boldsymbol{\Phi}$, by seeking the minimization of $\mathscr{H}_d(\boldsymbol{T})$ over a set of possible trees. Decision trees can, therefore, obtain the n-gram history equivalence classes, as a special case. Clearly, minimization of Eq. (15a) over all possible trees is not feasible. Instead, a top-down and greedy algorithm is typically used for decision tree development (Breiman et al., 1984): Starting from the initial tree $\boldsymbol{T}' = \{\boldsymbol{t}_0\}$, and until a stopping criterion is met, the algorithm performs the best split among the set of leaves $\boldsymbol{L}'$ of the current tree $\boldsymbol{T}'$, by means of question $\hat{\mathscr{Q}}(\hat{\boldsymbol{t}})$, which is determined as

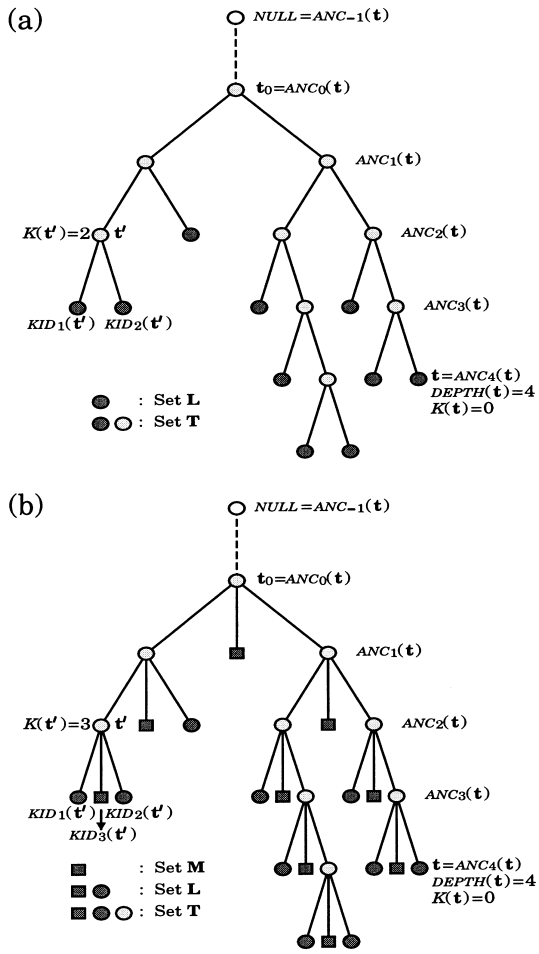

Fig. 2. (a) Notation for a binary decision tree (Section 3.2). (b) Notation for a ternary decision tree (Section 3.3).

$$\hat{\boldsymbol{t}} = \operatorname*{argmax}_{\boldsymbol{t} \in \boldsymbol{L}'} \Delta\mathscr{H}_d(\hat{\mathscr{Q}}(\boldsymbol{t})), \tag{16a}$$

where

$$\hat{\mathscr{Q}}(\boldsymbol{t}) = \operatorname*{argmax}_{\mathscr{Q}(\boldsymbol{t})} \Delta\mathscr{H}_d(\mathscr{Q}(\boldsymbol{t})) \tag{16b}$$

and

$$\Delta\mathscr{H}_d(\mathscr{Q}(\boldsymbol{t})) = f(\boldsymbol{t} \mid \boldsymbol{t}_0)\mathscr{H}_d(\boldsymbol{t}) - \sum_{i=1}^{K(\boldsymbol{t})} f(\mathrm{KID}_i(\boldsymbol{t}) \mid \boldsymbol{t}_0)\mathscr{H}_d(\mathrm{KID}_i(\boldsymbol{t})). \tag{16c}$$

There are three main issues related to decision tree design (Breiman et al., 1984): (a) the choice of candidate questions $\mathscr{Q}(\boldsymbol{t})$ in Eq. (16b); (b) the choice of the right size for the tree $\boldsymbol{T}$; and (c) the estimation of $\mathsf{Pr}[v \mid \boldsymbol{t}]$, where $\boldsymbol{v} \in \upsilon$ and $\boldsymbol{t} \in \boldsymbol{L}$.

In this work, we exclusively consider single feature questions of the type

$$\mathcal{Q}(\boldsymbol{t}) : \boldsymbol{h}_{n-1} = (\omega_{-1}, \omega_{-2}, \ldots, \omega_{-n+1}) \in \boldsymbol{\Phi}_{\boldsymbol{t}} \;\rightarrow\; \begin{cases} \mathrm{KID}_1(\boldsymbol{t}) & \text{if } \omega_{-j} \in A_1 \subset \upsilon, \\ \mathrm{KID}_2(\boldsymbol{t}) & \text{if } \omega_{-j} \in A_2 \subseteq \upsilon - A_1, \\ \mathrm{KID}_3(\boldsymbol{t}) & \text{if } \omega_{-j} \in \upsilon - \{A_1 \cup A_2\}, \end{cases} \tag{17}$$

for all $j = 1, 2, \ldots, n-1$, where $A_1, A_2 \neq \emptyset$ (clearly, $A_1 \cap A_2 = \emptyset$). We further restrict sets A_1 and A_2 to satisfy, either $A_1 \cup A_2 = \upsilon$, thus, resulting in binary splits ($\mathrm{KID}_3(\boldsymbol{t}) = \emptyset$), as in Section 3.2, or $A_1 \cup A_2 = \upsilon_j(\boldsymbol{t})$, where

$$\upsilon_j(\boldsymbol{t}) = \{\omega_{-j} : \boldsymbol{h}_{n-1} = (\omega_{-1}, \omega_{-2}, \ldots, \omega_{-n+1}) \in \boldsymbol{\Phi}_{\boldsymbol{t}}, \text{ with } C_d(\boldsymbol{h}_{n-1}) > 0\} \subseteq \upsilon, \tag{18}$$

thus, resulting in ternary splits, as in Section 3.3. Notice that, in general, $|\upsilon_j(\boldsymbol{t})| \gg 1$. Since there exist $2^{|\upsilon_j(\boldsymbol{t})|-1} - 1$ possible binary partitions of $\upsilon_j(\boldsymbol{t})$, solving Eq. (16b) is, in general, computationally infeasible. Two sub-optimal solutions to this problem are discussed in Sections 3.2 and 3.3. In this paper, multiple feature questions are not considered. Notice though, that both algorithms presented in Sections 3.2 and 3.3 can be readily extended to accommodate such questions, at a significantly higher computational cost.

With regard to determining the right size of a tree, and as a result of our experiments, we have concluded that it is beneficial to completely develop the tree using Eqs. (16a)–(16c) until $\Delta\mathcal{H}_d(\hat{\mathcal{Q}}(\hat{\boldsymbol{t}})) = 0$. In such a case, a simple *right-to-left* tree development algorithm can be used (Potamianos and Jelinek, 1997). We then employ a suitable smoothing algorithm, in order to determine $\mathsf{Pr}[v \mid \boldsymbol{t}]$, $v \in \upsilon$, $\boldsymbol{t} \in \boldsymbol{L}$, as we discuss in Section 3.4. Performance of the resulting language model is assessed by the minus log-probability on the test set, given by (compare to Eq. (2))

$$\mathcal{LP}_t(\boldsymbol{T}) = -\sum_{\boldsymbol{t} \in \boldsymbol{L}} \sum_{v \in \upsilon} \frac{C_t(v, \boldsymbol{t})}{C_t(\boldsymbol{t}_0)} \log_2 \mathsf{Pr}[v \mid \boldsymbol{t}].$$

3.2. Decision tree question design by means of vocabulary binary encoding

Let us assume that we have a unique binary representation of all $v \in \upsilon$, by means of a binary encoding of constant length L, $\underline{b}(v) = b_1(v)b_2(v)\ldots b_L(v)$, where $b_i(v) \in \{0, 1\}$, for all $i = 1, 2, \ldots, L$, and all $v \in \upsilon$. Such an encoding is typically obtained by means of the *mutual information clustering* tree (Brown et al., 1992b; Jelinek, 1997). As an example, in Table 2, we provide the binary encoding of our Brown corpus vocabulary. Notice now that sets $A_1 = \{v \in \upsilon : b_i(v) = 0\}$ and $A_2 = \{v \in \upsilon : b_i(v) = 1\}$ satisfy $A_1 \cup A_2 = \upsilon$, for all $i = 1, 2, \ldots, L$. Thus, the vocabulary binary encoding provides a means of partitioning set υ into $L < |\upsilon|$ different partitions within the totality $2^{|\upsilon|-1} - 1$ possible binary partitions of υ. We consider such partitions as our candidate questions of form (17) (see also Eq. (19)).

Given the vocabulary binary encoding, any history $\boldsymbol{h}_{n-1} = (\omega_{-1}, \omega_{-2}, \ldots, \omega_{-n+1})$ is equivalent to $\underline{b}(\boldsymbol{h}_{n-1}) = b_1(\omega_{-1})\ldots b_L(\omega_{-1})b_1(\omega_{-2})\ldots b_L(\omega_{-2})\ldots b_1(\omega_{-n+1})\ldots b_L(\omega_{-n+1})$, consisting of $L \times (n-1)$ bits. Thus, we can easily build a decision tree by using as possible questions (see also Eqs. (16a)–(16c) and (17))

$$\mathcal{Q}(\boldsymbol{t}, j, i) \colon \boldsymbol{h}_{n-1} = (\omega_{-1}, \omega_{-2}, \ldots, \omega_{-n+1}) \in \boldsymbol{\Phi}_{\boldsymbol{t}} \;\rightarrow\; \begin{cases} \mathrm{KID}_1(\boldsymbol{t}) & \text{if } b_i(\omega_{-j}) = 0, \\ \mathrm{KID}_2(\boldsymbol{t}) & \text{if } b_i(\omega_{-j}) = 1, \end{cases} \tag{19}$$

for all $j = 1, 2, \ldots, n-1$ and all $i = 1, 2, \ldots, L$; i.e., $\mathcal{Q}(\boldsymbol{t}, j, i)$ denotes the question at node $\boldsymbol{t}$, that concerns bit i of the binary encoding of ω_{-j}. Notice that questions (19) constitute a natural choice in the case $j = 1$, due to Eqs. (16a)–(16c) and the fact that the maximum mutual information criterion is used in the construction of the vocabulary clustering tree (Brown et al., 1992b).

In this paper, we consider two decision tree development algorithms based on questions (19). For any tree leaf $\boldsymbol{t}$, the first algorithm searches for the best question (19), over all $j = 1, 2, \ldots, n-1$, and all

Table 2
Binary encoding of our Brown corpus vocabulary, $\mathcal{V}$

v	$\underline{b}(v)$	v	$\underline{b}(v)$
!	00011110111000	O	00111111000000
#	10000000000000	P	01111010111000
$	00011110100000	Q	01111010101000
%	00011110111110	R	01111010001110
&	00011100000000	S	01111011100000
’	00110000000000	T	01111011000000
(	00011111000000	U	00111111100000
)	00011110111100	V	01111010000110
*	00011110000000	W	01111011111000
+	00011111100000	X	01111010101100
,	00011000000000	Y	01111010011100
-	01001110000000	Z	01111010000111
.	00000000000000	‘	00100000000000
0	00101000000000	a	11110000000000
1	00101100000000	b	01111110000000
2	00101111000000	c	01111100000000
3	00101111010000	d	01000000000000
4	00101111111000	e	11000000000000
5	00101111100000	f	01011000000000
6	00101111101000	g	01000100000000
7	00101111111100	h	01100000000000
8	00101111110000	i	11100000000000
9	00101110000000	j	01001111100000
:	00010000000000	k	01011101000000
;	00011110111111	l	01011110000000
?	00011110110000	m	01011100000000
A	00111100000000	n	01010000000000
B	01111010100000	o	11111000000000
C	01111011110000	p	01111111000000
D	01111010000000	q	01001111110000
E	00111110000000	r	01011111000000
F	01111010111100	s	01001100000000
G	01111010111110	t	01110000000000
H	01111010010000	u	11101000000000
I	00111000000000	v	01011101100000
J	01111010001000	w	01111000000000
K	01111010000100	x	01001111000000
L	01111010001100	y	01001000000000
M	01111010110000	z	01011101110000
N	01111010011000		

Symbol “#” denotes space.

$i = 1, 2, \ldots, L$, and, is, therefore, called *unrestricted*. The tree split induced by such best question is accepted, if the resulting tree entropy reduction is positive (see Eqs. (16a)–(16c)). We refer to this algorithm as ENC.1 (Potamianos and Jelinek, 1997). The second algorithm accepts, among questions (19), the first question that results in positive tree entropy reduction, while iterating within $j = 1, 2, \ldots, n-1$, and $i = 1, 2, \ldots, L$. It is, therefore, called *restricted*. Clearly, the second algorithm forces all possible questions (19) about ω_{-j} before any ones about $\omega_{-j-j'}$, where $j' \geqslant 1$. Thus, it mimics the construction of the n-gram language model. In addition, it favors questions about the most significant bits of the vocabulary binary encoding, i.e., questions (19) with smaller values of i. We refer to this algorithm as ENC.2 (Potamianos and Jelinek, 1997).

3.3. Decision tree question design by means of vocabulary K-means clustering

Let us again return to the problem of designing optimal questions (17). We first consider the problem in a more general setting. Given a tree node $\boldsymbol{t}$ and a history position $1 \leqslant j \leqslant n-1$, we would like to determine a K-way partition of sub-vocabulary $\upsilon_j(\boldsymbol{t})$, defined by Eq. (18), into sets $A_1, A_2, \ldots, A_K$, where $A_i \neq \emptyset$, $\cup_{i=1}^K A_i = \upsilon_j(\boldsymbol{t})$, and $A_i \cap A_{i'} = \emptyset$, for $i \neq i'$, such that the $(K+1)$-way split of node $\boldsymbol{t}$ into new nodes $\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_K$, and a middle node $\boldsymbol{t}'$ ($\boldsymbol{t}' \cup_{i=1}^K \boldsymbol{t}_i = \boldsymbol{t}$), by means of question

$$\mathscr{Q}(\boldsymbol{t}; A_1, \ldots, A_{K-1}) : \boldsymbol{h}_{n-1} = (\omega_{-1}, \ldots, \omega_{-n+1}) \in \boldsymbol{\Phi}_{\boldsymbol{t}} \;\rightarrow\; \begin{cases} \boldsymbol{t}_i = \mathrm{KID}_i(\boldsymbol{t}) & \text{if } \omega_{-j} \in A_i, \\ & i = 1, 2, \ldots, K, \\ \boldsymbol{t}' = \mathrm{KID}_{K+1}(\boldsymbol{t}) & \text{if } \omega_{-j} \in \upsilon - \upsilon_j(\boldsymbol{t}), \end{cases} \tag{20}$$

is optimal. Thus, we seek partition (see also Eqs. (16a)–(16c))

$$(\hat{A}_1, \hat{A}_2, \ldots, \hat{A}_K) = \underset{(A_1, A_2, \ldots, A_K): \cup_{i=1}^K A_i = \upsilon_j(\boldsymbol{t}); A_i \neq \emptyset; A_i \cap A_{i'} = \emptyset}{\arg\max} \left\{ \mathscr{H}_d(\boldsymbol{t}) - \sum_{i=1}^K f(\boldsymbol{t}_i \mid \boldsymbol{t}) \mathscr{H}_d(\boldsymbol{t}_i) \right\}. \tag{21}$$

Let us define *atoms*, or, *atomic nodes*, $\boldsymbol{v}$, for all $v \in \upsilon_j(\boldsymbol{t})$ (see Eq. (18)), as

$$\boldsymbol{v} = \{\boldsymbol{h}_{n-1} = (\omega_{-1}, \omega_{-2}, \ldots, \omega_{-n+1}) \in \boldsymbol{\Phi}_{\boldsymbol{t}}\colon\; C_d(\boldsymbol{h}_{n-1}) > 0, \text{ and } \omega_{-j} = v\}.$$

Clearly, for every such atomic node, a probability mass function

$$f(\omega \mid \boldsymbol{v}) = \frac{C_d(\omega, \boldsymbol{v})}{C_d(\boldsymbol{v})} \quad \text{for all } \omega \in \upsilon,$$

can be defined (see also Eq. (14)). Notice now that Eq. (21) is equivalent to (Chou, 1991)

$$(\hat{A}_1, \hat{A}_2, \ldots, \hat{A}_K) = \underset{(A_1, A_2, \ldots, A_K): \cup_{i=1}^K A_i = \upsilon_j(\boldsymbol{t}); A_i \neq \emptyset; A_i \cap A_{i'} = \emptyset}{\arg\min} \left\{ \sum_{i=1}^K f(\boldsymbol{t}_i \mid \boldsymbol{t}) \mathscr{H}_d(\boldsymbol{t}_i) - \sum_{v \in \upsilon_j(\boldsymbol{t})} f(\boldsymbol{v} \mid \boldsymbol{t}) \mathscr{H}_d(\boldsymbol{v}) \right\}. \tag{22}$$

Clearly, the argument in Eq. (22) equals (see also Eq. (15b))

$$\sum_{i=1}^K \sum_{v \in A_i} f(\boldsymbol{v} \mid \boldsymbol{t}) \sum_{\omega \in \upsilon} f(\omega \mid \boldsymbol{v}) \log_2 \frac{f(\omega \mid \boldsymbol{v})}{f(\omega \mid \boldsymbol{t}_i)}.$$

We now re-partition node $\boldsymbol{t}$ into $\{\boldsymbol{t}'_1, \boldsymbol{t}'_2, \ldots, \boldsymbol{t}'_K\}$, defined by

$$\boldsymbol{t}'_i = \{\boldsymbol{h}_{n-1} = (\omega_{-1}, \omega_{-2}, \ldots, \omega_{-n+1}) \in \boldsymbol{\Phi}_{\boldsymbol{t}}\colon\; C_d(\boldsymbol{h}_{n-1}) > 0, \text{ and } \omega_{-j} \in A'_i\}, \tag{23a}$$

where sets

$$A'_i = \left\{ v \in \upsilon_j(\boldsymbol{t})\colon\; i = \arg\min_{k \in [1,K]} \sum_{\omega \in \upsilon} f(\omega \mid \boldsymbol{v}) \log_2 \frac{f(\omega \mid \boldsymbol{v})}{f(\omega \mid \boldsymbol{t}_k)} \right\}, \tag{23b}$$

for $i = 1, 2, \ldots, K$, partition set $\upsilon_j(\boldsymbol{t})$. The new partition is "better" in terms of Eq. (22), since (Chou, 1991)

$$\sum_{i=1}^K \sum_{v \in A'_i} f(\boldsymbol{v} \mid \boldsymbol{t}) \sum_{\omega \in \upsilon} f(\omega \mid \boldsymbol{v}) \log_2 \frac{f(\omega \mid \boldsymbol{v})}{f(\omega \mid \boldsymbol{t}'_i)} \;\leqslant\; \sum_{i=1}^K \sum_{v \in A_i} f(\boldsymbol{v} \mid \boldsymbol{t}) \sum_{\omega \in \upsilon} f(\omega \mid \boldsymbol{v}) \log_2 \frac{f(\omega \mid \boldsymbol{v})}{f(\omega \mid \boldsymbol{t}_i)}.$$

The new partition *centroids* (i.e., the probability mass functions of the resulting nodes) are given by

$$f(\omega \mid \boldsymbol{t}'_i) = \frac{\sum_{v \in A'_i} f(\omega \mid \boldsymbol{v}) f(\boldsymbol{v} \mid \boldsymbol{t})}{\sum_{v \in A'_i} f(\boldsymbol{v} \mid \boldsymbol{t})}, \tag{24}$$

for all $\omega \in \upsilon$, and $i = 1, 2, \ldots, K$. Eqs. (23a), (23b) and (24) constitute *Chou's partitioning algorithm*, an iterative technique for improving on an initial partition $A_1, A_2, \ldots, A_K$, of $\upsilon_j(\boldsymbol{t})$ on basis of Eq. (22): until no further entropy improvement occurs by split (20), repeat the following two steps: first, compute the centroids of the current partition by means of Eq. (24) (replace A_i' by A_i), and, second, re-partition $\upsilon_j(\boldsymbol{t})$ according to Eq. (23b). Notice that the algorithm results in a locally optimal partition (22) (Chou, 1991).

There exists a subtle difficulty, when repartitioning a current partition by means of Eq. (23b). It may happen that for some current node $\boldsymbol{t}_k$, atom $\boldsymbol{v}$, and $\omega \in \upsilon$, the following holds: $f(\omega \mid \boldsymbol{v}) > 0$ and $f(\omega \mid \boldsymbol{t}_k) = 0$. The *divergence* in Eq. (23b) then becomes infinite. To avoid smoothing the partition centroids (24), as is suggested by Chou (1991), we consider Chou's partitioning algorithm specialized to the *Gini index of diversity* (Breiman et al., 1984; Chou, 1991), rather than to the entropy. The Gini index of diversity is given by (compare to Eq. (15b))

$$\mathscr{G}_d(\boldsymbol{t}) = 1 - \sum_{v \in \upsilon} f(v \mid \boldsymbol{t})^2 = \sum_{v \in \upsilon} f(v \mid \boldsymbol{t})[1 - f(v \mid \boldsymbol{t})].$$

In such a case, the *euclidean distance* between probability mass functions $f(\omega \mid \boldsymbol{v})$ and $f(\omega \mid \boldsymbol{t}_k)$ replaces their divergence in Eq. (23b) (Potamianos and Jelinek, 1997).

In this paper, we employ Chou's partitioning algorithm for the binary ($K = 2$) partitioning of set $\upsilon_j(\boldsymbol{t})$. The candidate questions then become (see Eqs. (17) and (20))

$$\mathscr{Q}(\boldsymbol{t}, j, A_1) : \boldsymbol{h}_{n-1} = (\omega_{-1}, \ldots, \omega_{-n+1}) \in \boldsymbol{\Phi}_{\boldsymbol{t}} \quad \rightarrow \quad \begin{cases} \mathrm{KID}_1(\boldsymbol{t}) & \text{if } \omega_{-j} \in A_1 \subset \upsilon_j(\boldsymbol{t}), \\ \mathrm{KID}_2(\boldsymbol{t}) & \text{if } \omega_{-j} \in A_2 = \upsilon_j(\boldsymbol{t}) - A_1, \\ \mathrm{KID}_3(\boldsymbol{t}) & \text{if } \omega_{-j} \in \upsilon - \upsilon_j(\boldsymbol{t}). \end{cases} \tag{25}$$

Since Chou's partitioning algorithm is locally only optimal, we use multiple runs of the algorithm by starting from various initial partitions, constructed by partitioning $\upsilon_j(\boldsymbol{t})$ at random. The entropy is used as a figure of merit in comparing the resulting sub-optimal splits (Potamianos and Jelinek, 1997).

We consider three versions of Chou's partitioning algorithm for decision tree design. The first one, referred to as *unrestricted*, or CHOU.1, at every candidate leaf, considers questions $\mathscr{Q}(\boldsymbol{t}, j, \hat{A}_1)$, where sets $\hat{A}_1$ result from Chou's partitioning algorithm, for all $j = 1, 2, 3, \ldots, n-1$. Thus, at every candidate leaf, $n-1$ candidate questions (splits) are considered, and the one that maximizes the induced (by the split) tree entropy reduction is chosen (see also Eqs. (16a)–(16c)). The second algorithm, referred to as *restricted*, or CHOU.2, at every candidate leaf, picks as the candidate question the first one that results in $\Delta\mathscr{H}_d(\mathscr{Q}(\boldsymbol{t}, j, \hat{A}_1)) > 0$, during iteration $j = 1, 2, 3, \ldots, n-1$. It is, therefore, biased towards asking questions about ω_{-j} before asking questions about $\omega_{-j-j'}$, where $j' \geqslant 1$. Finally, the third algorithm, referred to as *n-gram equivalent*, or CHOU.3, creates a tree that contains all n-gram tree nodes in it. At every candidate leaf $\boldsymbol{t}$, the algorithm picks as the candidate question the first one that satisfies either $\Delta\mathscr{H}_d(\mathscr{Q}(\boldsymbol{t}, j, \hat{A}_1)) > 0$, or $\Delta\mathscr{H}_d(\mathscr{Q}(\boldsymbol{t}, j, \hat{A}_1)) = 0$ and $\mid \upsilon_j(\boldsymbol{t}) \mid > 1$, during iteration $j = 1, 2, 3, \ldots, n-1$. Notice that the second condition arises, whenever $\mid \upsilon_j(\boldsymbol{t}) \mid > 1$, and $f(\omega \mid \boldsymbol{v}) = f(\omega \mid \boldsymbol{v}')$, for all $\omega \in \upsilon$, and all $v, v' \in \upsilon_j(\boldsymbol{t})$. In such a case, any question (25) induces zero tree entropy reduction, since $\mathscr{H}_d(\boldsymbol{t}) = \mathscr{H}_d(\mathrm{KID}_1(\boldsymbol{t})) = \mathscr{H}_d(\mathrm{KID}_2(\boldsymbol{t})) = \mathscr{H}_d(\boldsymbol{v})$, for all $v \in \upsilon_j(\boldsymbol{t})$ (see also Eqs. (15b) and (21)). We clearly have to "force" a question (25) (here, we pick a random partition of $\upsilon_j(\boldsymbol{t})$), in order to ensure that all n-gram nodes are contained in the resulting tree. Such nodes satisfy $\upsilon_j(\boldsymbol{t}) = 1$ before any questions about $\omega_{-j-j'}$, where $j' \geqslant 1$, are asked (Potamianos and Jelinek, 1997).

3.4. Decision tree smoothing

In Sections 3.2 and 3.3, we discussed algorithms for decision tree design, based on the development data set. Clearly, every leaf $\boldsymbol{t} \in \boldsymbol{L} - \boldsymbol{M}$ can be assigned maximum likelihood estimates of probabilities (1), given

by $\mathsf{Pr}_{\mathsf{ML}}[v \mid \boldsymbol{t}]$, in Eq. (14). Of course, such estimates do not generalize to unseen data well, and $\mathscr{LP}_t(\boldsymbol{T})$, most certainly, becomes infinite. Therefore, smoothing techniques must be used in conjunction with Eq. (14), in order to provide the required generalization.

In Section 2, we considered a variety of smoothing techniques applied to the n-gram letter language model. We concluded that, among these techniques, the bottom–up deleted interpolation is the most robust to sparseness of training data, outperforming the rest. We, therefore, proceed to smooth our decision tree leaf probability mass functions by means of this algorithm.

Revisiting Eqs. (12a)–(12e), we discover that decision trees exhibit certain peculiarities that make the straightforward application of the bottom–up deleted interpolation algorithm rather difficult: decision trees are not balanced, thus, in general, tree leaves have different numbers of ancestors, some of the leaves often being of very small or very large depth. Therefore, when trying to simultaneously smooth the probability mass functions of the nodes in set $\boldsymbol{L} - \boldsymbol{M}$ (i.e., of all tree leaves that are not middle nodes), as we do in the n-gram case, we face the obvious difficulty of how to define an equal number of distinct ancestors for each one of these leaves. In addition, the ternary decision trees, discussed in Section 3.3, pose one more challenge: at every split (25), a middle node is defined. Histories of both the held-out and test sets might end up in some of the middle nodes, which, therefore, should have a probability mass function associated with them. We choose to assign to them the probability mass function of their parent node, i.e.,

$$\mathsf{Pr}[v \mid \boldsymbol{t}] = \mathsf{Pr}[v \mid \mathrm{PAR}(\boldsymbol{t})] \quad \text{for all } v \in \upsilon, \text{ all } \boldsymbol{t} \in \boldsymbol{M}.$$

Practically, therefore, smoothed probability mass functions of all tree nodes are needed. Next, we propose an algorithm that addresses these issues.

Let us first define

$$\mathrm{DEPTH}_{\mathsf{min}} = \min\{\mathrm{DEPTH}(\boldsymbol{t}) : \boldsymbol{t} \in \boldsymbol{L} - \boldsymbol{M}\} \geqslant 0$$

and

$$\mathrm{DEPTH}_{\mathsf{max}} = \max\{\mathrm{DEPTH}(\boldsymbol{t}) : \boldsymbol{t} \in \boldsymbol{L} - \boldsymbol{M}\} \leqslant (n-1)\,(|\upsilon| - 1),$$

and consider some integer D: $0 \leqslant D \leqslant \mathrm{DEPTH}_{\mathsf{max}}$. We then partition the set of tree leaves (excluding middle nodes) $\boldsymbol{L} - \boldsymbol{M}$ into sets

$$\boldsymbol{S}_1 = \{\boldsymbol{t} \in \boldsymbol{L} - \boldsymbol{M}\colon \mathrm{DEPTH}(\boldsymbol{t}) < D\} \quad \text{and} \quad \boldsymbol{S}_2 = \{\boldsymbol{t} \in \boldsymbol{L} - \boldsymbol{M}\colon \mathrm{DEPTH}(\boldsymbol{t}) \geqslant D\},$$

and the set of internal tree nodes $\boldsymbol{T} - \boldsymbol{L}$ into sets

$$\boldsymbol{S}_3 = \{\boldsymbol{t} \in \boldsymbol{T} - \boldsymbol{L}\colon \mathrm{DEPTH}(\boldsymbol{t}) < D\} \quad \text{and} \quad \boldsymbol{S}_4 = \{\boldsymbol{t} \in \boldsymbol{T} - \boldsymbol{L}\colon \mathrm{DEPTH}(\boldsymbol{t}) \geqslant D\}.$$

Clearly, $\boldsymbol{S}_1 \cup \boldsymbol{S}_2 = \boldsymbol{L} - \boldsymbol{M}$, $\boldsymbol{S}_1 \cap \boldsymbol{S}_2 = \emptyset$, $\boldsymbol{S}_3 \cup \boldsymbol{S}_4 = \boldsymbol{T} - \boldsymbol{L}$ and $\boldsymbol{S}_3 \cap \boldsymbol{S}_4 = \emptyset$. For binary trees, we need smoothed probability mass functions for all nodes in $\boldsymbol{S}_1 \cup \boldsymbol{S}_2$, whereas, for ternary trees, we need smoothed estimates for all tree nodes. Clearly, for every node in $\boldsymbol{S}_2 \cup \boldsymbol{S}_4$, we can define $D + 2$ distinct ancestor nodes of it (see Eqs. (13a) and (13b)). We can, therefore, easily apply the bottom-up deleted interpolation algorithm on the node set $\boldsymbol{S}_2 \cup \boldsymbol{S}_4$. Since this constitutes the most robust smoothing algorithm, we would like set $\boldsymbol{S}_2 \cup \boldsymbol{S}_4$ to contain as many nodes as possible. On the other hand, we would like value D to be large enough, so that enough "levels" of smoothing can be used in the deleted interpolation algorithm. In our experiments, we use value $D = \mathrm{DEPTH}_{\mathsf{min}}$. Clearly, under this choice of D, $\boldsymbol{S}_1 = \emptyset$.

Once the value of D is chosen, we proceed to smooth the probability mass functions of nodes $\boldsymbol{t} \in \boldsymbol{S}_1 \cup \boldsymbol{S}_3$. The depth of such nodes varies from zero to $D - 1$, therefore, a robust deleted interpolation algorithm cannot be used. Our experimental results in Section 2 (see also Table 1) indicate that a powerful and simple smoothing algorithm, that does not require node bucketing for optimization of coefficients, is the back-off method (8a)–(8c) in conjunction with the discounting by half law of succession (4b). Thus, we set $\mathsf{Pr}[v \mid \boldsymbol{t}] = \mathsf{Pr}_{\mathsf{BOF.2}}[v \mid \boldsymbol{t}]$, for all $v \in \upsilon$, and $\boldsymbol{t} \in \boldsymbol{S}_1 \cup \boldsymbol{S}_3$, where (see Eqs. (4b), (8a)–(8c), (13a), (13b) and (14))

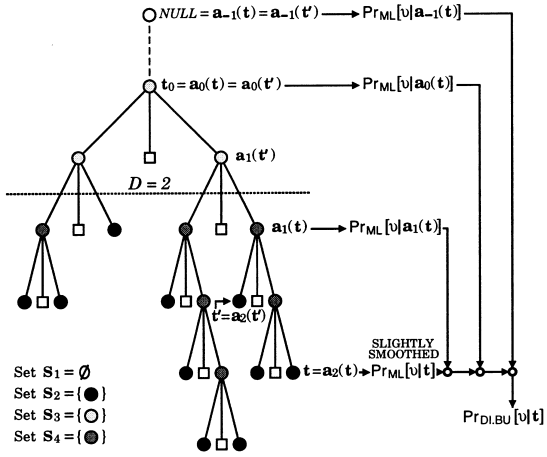

Fig. 3. Tree smoothing by means of the bottom-up deleted interpolation algorithm. In this case, $D = 2$.

$$\mathsf{Pr}_{\mathsf{BOF.2}}[v \mid \mathrm{ANC}_{-1}(\boldsymbol{t})] = \frac{1}{|\,\upsilon\,|},$$

and, for $i = 0, 1, 2, \ldots, \mathrm{DEPTH}(\boldsymbol{t})$,

$$\mathsf{Pr}_{\mathsf{BOF.2}}[v \mid \mathrm{ANC}_i(\boldsymbol{t})] = \begin{cases} \frac{C_d(v,\mathrm{ANC}_i(\boldsymbol{t}))}{C_d(\mathrm{ANC}_i(\boldsymbol{t}))} & \text{if } q(\mathrm{ANC}_i(\boldsymbol{t})) = |\,\upsilon\,|; \text{otherwise:} \\ \frac{C_d(v,\mathrm{ANC}_i(\boldsymbol{t}))-0.5}{C_d(\mathrm{ANC}_i(\boldsymbol{t}))} & \text{if } C_d(v, \mathrm{ANC}_i(\boldsymbol{t})) > 0, \\ \beta_i(\boldsymbol{t})\mathsf{Pr}_{\mathsf{BOF.2}}[v \mid \mathrm{ANC}_{i-1}(\boldsymbol{t})] & \text{if } C_d(v, \mathrm{ANC}_i(\boldsymbol{t})) = 0, \end{cases}$$

where, if $q(\mathrm{ANC}_i(\boldsymbol{t})) < |\,\upsilon\,|$,

$$\beta_i(\boldsymbol{t}) = \frac{1 \;-\sum_{v:\; C_d(v,\mathrm{ANC}_i(\boldsymbol{t}))>0} \frac{C_d(v,\mathrm{ANC}_i(\boldsymbol{t}))-0.5}{C_d(\boldsymbol{t})}}{\sum_{v:\; C_d(v,\mathrm{ANC}_i(\boldsymbol{t}))=0} \mathsf{Pr}_{\mathsf{BOF.2}}[v \mid \mathrm{ANC}_{i-1}(\boldsymbol{t})]}.$$

For the remaining nodes $\boldsymbol{t} \in \boldsymbol{S}_2 \cup \boldsymbol{S}_4$, we use the bottom-up deleted interpolation algorithm, with $D+1$ iterations. For this purpose, we define $D+2$ ancestor nodes of $\boldsymbol{t}$, by (see also Eqs. (12a)–(12e) and (13b), and Fig. 3) [8]

$$\boldsymbol{a}_{-1}(\boldsymbol{t}) = \mathrm{NULL} \quad \text{and} \quad \boldsymbol{a}_i(\boldsymbol{t}) = \mathrm{ANC}_{\lfloor i\ \mathrm{DEPTH}(\boldsymbol{t})/D \rfloor}(\boldsymbol{t}) \quad \text{for} \quad i = 0, 1, 2, \ldots, D.$$

Clearly, $\boldsymbol{a}_0(\boldsymbol{t}) = \boldsymbol{t}_0$, $\boldsymbol{a}_D(\boldsymbol{t}) = \boldsymbol{t}$ and $\mathrm{DEPTH}(\boldsymbol{a}_{i+1}(\boldsymbol{t})) - \mathrm{DEPTH}(\boldsymbol{a}_i(\boldsymbol{t})) \simeq \mathrm{DEPTH}(\boldsymbol{t})/D$, for $i = 0, 1, \ldots, D-1$. Then, we seek to find smoothing coefficients $\lambda^{(i)}(\boldsymbol{t})$, $i = -1, 0, 1, \ldots, D$, that satisfy

$$0 < \lambda^{(i)}(\boldsymbol{t}) < 1 \quad \text{for } i = -1, 0, 1, 2, \ldots, D, \text{ and } \sum_{i=-1}^{D} \lambda^{(i)}(\boldsymbol{t}) = 1, \tag{26a}$$

such that the probability mass function

$$\mathsf{Pr}_{\mathsf{DI}}[v \mid \boldsymbol{t}] = \sum_{i=-1}^{D} \lambda^{(i)}(\boldsymbol{t}) \mathsf{Pr}_{\mathsf{ML}}[v \mid \boldsymbol{a}_i(\boldsymbol{t})] \tag{26b}$$

[8] We denote $\lfloor x \rfloor = \max\{i : i \text{ is integer, and } i \leqslant x\}$.

generalizes to held-out data set well. Notice the similarity between Eqs. (9a), (9b), (26a), (26b). The bottom–up smoothing algorithm (12a)–(12e) can be now applied in a straightforward manner for decision tree smoothing (see also Fig. 3).

3.5. Experimental results

We now compare the performance of five decision tree based letter language models and the baseline n-gram letter language model on the Brown corpus. The decision trees are trained on n-gram history data, $1 \leqslant n \leqslant 10$, by means of the unrestricted and restricted history encoding algorithms (ENC.1 and ENC.2), discussed in Section 3.2, and the unrestricted, restricted, and n-gram equivalent Chou's partitioning algorithms (CHOU.1, CHOU.2, and CHOU.3), presented in Section 3.3. All trees are completely developed and smoothed, as discussed in Section 3.4.

In Table 3, we depict the minus log-probabilities of the six language models on the test subset of the Brown corpus. Clearly, the baseline n-gram language model defeats the first four decision tree language models. Notice that the performance of the unrestricted ENC.1 and CHOU.1 models deteriorates for $n > 8$ and $n > 7$, respectively, with the CHOU.1 algorithm giving a better result. The performance of the restricted algorithms (ENC.2 and CHOU.2) further favors Chou's partitioning algorithm. Notice that only the decision trees developed by means of the CHOU.3 algorithm are equivalent to n-grams, although (having more internal nodes) they offer additional freedom in the smoothing algorithm. This translates into a slightly better performance of the CHOU.3 language model over the n-gram.

3.6. Remarks on decision tree based language modeling

The experimental results reported above are disappointing. The decision tree language models (with the exception of the CHOU.3 one) clearly fail to improve on the baseline n-gram language model. The cause seems to be data sparseness, coupled with the fact that the decision tree development algorithms decide on the tree splits solely on basis of seen data. Whenever, for example, a question of type (17) is asked, there might exist test set histories that correspond to set $\upsilon - \upsilon_j(\boldsymbol{t})$. In Eq. (19), such histories drop to the left or right children nodes depending on the highly irrelevant vocabulary encoding, whereas, in Eq. (25), they end up in the middle node, and are assigned the probability of the parent node $\boldsymbol{t}$. Both treatments are clearly inadequate, although our results indicate that the second is preferable. An n-gram language model, however, deals with unseen histories in a very natural way. It considers the largest length seen n'-gram ($n' < n$), and it assigns its probability mass function to the unseen n-gram. In the case of a decision tree

Table 3
Minus log-probability on our Brown corpus test subset, $\mathscr{LP}_t$, of the n-gram and five decision tree based language models (smoothed by means of bottom-up deleted interpolation, for $n = 1, 2, \ldots, 10$)

n	n-gram	ENC.1	ENC.2	CHOU.1	CHOU.2	CHOU.3
1	4.380	4.380	4.380	4.380	4.380	4.380
2	3.470	3.470	3.470	3.470	3.470	3.470
3	2.850	2.850	2.850	2.850	2.850	2.850
4	2.326	2.327	2.326	2.326	2.325	2.325
5	2.007	2.012	2.010	2.008	2.006	2.004
6	1.878	1.893	1.889	1.887	1.879	1.872
7	1.831	1.861	1.855	**1.853**	1.839	1.825
8	1.811	**1.857**	1.848	1.856	1.827	1.807
9	1.801	1.858	1.844	1.866	1.820	1.800
10	**1.796**	1.862	**1.842**	1.879	**1.815**	**1.795**

All results are measured in bits per character.

middle node, its probability mass function typically corresponds to a cluster of histories that apparently have less predictive capability than a specific character string (n'-gram). It is, therefore, not surprising that, when restricting the decision trees to mimic the hierarchical structure of n-grams (such as the ENC.2, CHOU.2, and CHOU.3 algorithms), the decision tree results on test data improve.

4. Summary and future work

In this work, we provided a comparative study of n-gram and decision tree language models, by conducting letter language modeling experiments on the Brown corpus. We first studied a variety of language model smoothing algorithms and compared their performance on the n-gram letter language model. We concluded that the bottom–up deleted interpolation algorithm constitutes the most robust smoothing technique. We then considered two algorithms for decision tree question design and concluded that the one based on K-means clustering is preferable. However, when comparing the n-gram and decision tree letter language model performance on the test set, and for large n, the former outperforms the latter. This, we believe, is due to the fact that character strings are naturally well modeled with the hierarchical structure of n-grams.

It is of great interest to consider the relative merits of the various algorithms discussed in this paper, in the case of large vocabulary word language modeling. As stated in the introduction, the data sparseness problem rapidly arises in such a case, as n increases. We believe that the employment of certain types of class language models can alleviate this problem and thus mimic the gradual appearance of data sparseness that we encountered in the letter language models. We expect our conclusions on smoothing algorithms to generalize to such cases. In addition, we believe that multiple feature questions can be beneficial for decision tree word language model design. We plan to investigate these issues in the future.

Acknowledgements

We would like to thank Eric Ristad for his feedback on the technical report version of this paper (Potamianos and Jelinek, 1997), and Mari Ostendorf for inspiring discussions on decision tree algorithms. In addition, we would like to thank Xiaoqiang Luo for helping with Brown corpus data preparation. We also wish to thank four anonymous reviewers for their constructive comments on an earlier version of this article.

References

Bahl, L.R., Jelinek, F., Mercer, R.L., 1983. A maximum likelihood approach to continuous speech recognition. IEEE Trans. Pattern Anal. Machine Intell. 5, 179–190.

Bahl, L.R., Brown, P.F., deSouza, P.V., Mercer, R.L., 1989. A tree-based statistical language model for natural language speech recognition. IEEE Trans. Acoust. Speech Signal Process. 37, 1001–1008.

Bahl, L.R., Brown, P.F., deSouza, P.V., Mercer, R.L., Nahamoo, D., 1991. A fast algorithm for deleted interpolation. In: Proc. Europ. Conf. Speech Comm. Tech., Genova, pp. 1209–1212.

Bell, T.C., Cleary, J.G., Witten, I.H., 1990. Text Compression. Prentice-Hall, Englewood Cliffs, NJ.

Betz, M., Hild, H., 1995. Language models for a spelled letter recognizer. In: Proc. Internat. Conf. Acoust. Speech Signal Process., Detroit, pp. 856–859.

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., 1984. Classification and Regression Trees. Chapman and Hall, New York.

Brown, P.F., Della Pietra, S.A., Della Pietra, V.J., Lai, J.C., Mercer, R.L., 1992a. An estimate of an upper bound for the entropy of English. Comput. Linguistics 18, 31–40.

Brown, P.F., Della Pietra, V.J., deSouza, P.V., Lai, J.C., Mercer, R.L., 1992b. Class-based *n*-gram models of natural language. Comput. Linguistics 18, 467–479.

Chen, S.F., Goodman, J., 1996. An empirical study of smoothing techniques for language modeling. In: Proc. 34th Ann. Meeting Assoc. Comput. Linguistics, Santa Cruz, pp. 310–318.

Chou, P.A., 1991. Optimal partitioning for classification and regression trees. IEEE Trans. Pattern Anal. Machine Intell. 13, 340–354.

Efron, B., 1982. The jackknife, the bootstrap and other resampling plans. In: CBMS-NSF Reg. Conf. Ser. Appl. Math., SIAM, Philadelphia, PA.

Good, I.J., 1953. The population frequencies of species and the estimation of population parameters. Biometrika 40, 237–264.

Jelinek, F., 1997. Statistical Methods for Speech Recognition. MIT Press, Cambridge, MA.

Jelinek, F., Mercer, R.L., 1980. Interpolated estimation of Markov source parameters from sparse data. In: Gelsema, E.S., Kanal, L.N. (Eds.), Pattern Recognition in Practice. North-Holland, Amsterdam, pp. 381–397.

Katz, S.M., 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. IEEE Trans. Acoust. Speech Signal Process. 35, 400–401.

Kucera, H., Francis, W., 1967. Computational Analysis of Present-Day American English. Brown Univ. Press, Providence, RI.

Ney, H., Essen, U., Kneser, R., 1995. On the estimation of 'small' probabilities by leaving-one-out. IEEE Trans. Pattern Anal. Machine Intell. 17, 1202–1212.

Potamianos, G., Jelinek, F., 1997. A study of n-gram and decision tree letter language modeling methods. CLSP Research Notes 13, Center for Language and Speech Processing, The Johns Hopkins University, Baltimore, MD.

Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T., 1988. Numerical Recipes in C. The Art of Scientific Computing. Cambridge Univ. Press, Cambridge.

Ristad, E.S., 1995. A natural law of succession. Research Report CS-TR-495-95, Department of Computer Science, Princeton University, Princeton, NJ.

Ristad, E.S., Thomas, R.G., 1995. New techniques for context modeling. In: Proc. 33rd Ann. Meeting Assoc. Comput. Linguistics, Cambridge, pp. 220–227.

Ristad, E.S., Thomas, R.G., 1997a. Hierarchical non-emitting Markov models. In: Proc. 35th Ann. Meeting Assoc. Comput. Linguistics, Madrid, pp. 381–385.

Ristad, E.S., Thomas, R.G., 1997b. Hierarchical non-emitting Markov models. Research Report CS-TR-544-97, Department of Computer Science, Princeton University, Princeton, NJ.

Shannon, C.E., 1951. Prediction and entropy of printed English. Bell Syst. Tech. J. 30, 50–64.