Available online at www.sciencedirect.com
ScienceDirect
Speech Communication 50 (2008) 434–451
SPEECH COMMUNICATION
www.elsevier.com/locate/specom

Joint-sequence models for grapheme-to-phoneme conversion

Maximilian Bisani *, Hermann Ney

Lehrstuhl für Informatik VI, RWTH Aachen University, Ahornstraße 55, D-52056 Aachen, Germany

Received 14 March 2007; received in revised form 11 January 2008; accepted 14 January 2008

Abstract

Grapheme-to-phoneme conversion is the task of finding the pronunciation of a word given its written form. It has important applications in text-to-speech and speech recognition. Joint-sequence models are a simple and theoretically stringent probabilistic framework that is applicable to this problem. This article provides a self-contained and detailed description of this method. We present a novel estimation algorithm and demonstrate high accuracy on a variety of databases. Moreover, we study the impact of the maximum approximation in training and transcription, the interaction of model size parameters, *n*-best list generation, confidence measures, and phoneme-to-grapheme conversion. Our software implementation of the method proposed in this work is available under an Open Source license.

Keywords: Grapheme-to-phoneme; Letter-to-sound; Phonemic transcription; Joint-sequence model; Pronunciation modeling

1. Introduction

Alphabetic writing systems are based on the idea that the orthographic form is a conventional representation of a word's pronunciation. In a perfectly phonological alphabet, there would be a one-to-one correspondence between letters (graphemes) and phonemes. However, in most natural languages the association between letters and sounds is to some extent ambiguous and context dependent. Most languages have continued to evolve after their orthographies have been canonized, so that the strict correspondence between letters and sounds has weakened over time. In particular, loanwords often retain the spelling of their language of origin instead of being adapted to match their host language's orthographic conventions.

Grapheme-to-phoneme conversion (G2P) refers to the task of finding the pronunciation of a word given its written form. It has important applications in human language technologies, especially speech synthesis, but also speech recognition and sounds-like queries in textual databases.

The main contributions of the present work are:

- A coherent approach to grapheme-to-phoneme conversion is presented that is well founded on statistical decision theory.
- Several parameters and variations of this method are studied systematically that have not been addressed comprehensively in previous publications. In particular, different alignment schemata, model smoothing, as well as the use of maximum approximations in training and application are studied. Moreover, we provide details of our implementation, which is freely available.
- It is demonstrated that the proposed method performs more accurately than or on par with all previously published results on several test sets.

In this article, we will first attempt an overview of the variety of published grapheme-to-phoneme conversion techniques in Section 2. In the remainder we will focus on an approach using statistical joint-sequence models. After laying the theoretical foundations of this approach in Section 3, we will undertake a detailed exposition of this

* Corresponding author. Present address: Nuance Communications Inc., 1 Wayside Road, Burlington, MA 01803, USA. Tel.: +1 781 565 5000; fax: +1 781 565 5001.
E-mail addresses: bisani@informatik.rwth-aachen.de (M. Bisani), ney@informatik.rwth-aachen.de (H. Ney).

0167-6393/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.specom.2008.01.002

method in Sections 4–6. Thereby, we will introduce a novel model estimation technique and will discuss several implementation aspects which have not been addressed in previous publications. Section 7 presents experimental results demonstrating the accuracy of the proposed method, which will be analyzed in the Section 8. Finally, in Section 9 we will discuss some of our experiences with using grapheme-to-phoneme conversion in practical applications.

2. Review of grapheme-to-phoneme conversion techniques

Automatic grapheme-to-phoneme conversion was first considered in the context of text-to-speech (TTS) applications. After normalization (expanding abbreviation, numerals, etc.) the input text needs to be converted to a sequence of phonemes which is then used to control a speech synthesizer. The simplest technique is *dictionary look-up*. While effective, it has serious limitations: making a pronunciation dictionary of significant size (over 100,000 entries) by hand is tedious and therefore costly. Also the storage requirements of such a database can be problematic for embedded or mobile devices. More importantly, a finite dictionary will always have limited coverage, while TTS systems are often expected to handle arbitrary words.

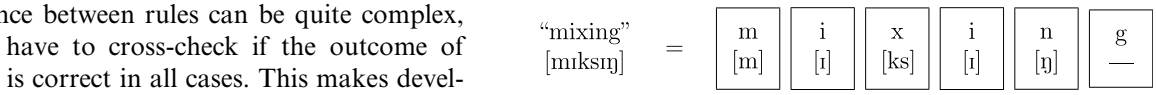
To overcome the limitations of simple dictionary look-up, *rule-based* conversion systems were developed. These can typically be formulated in the framework of finite-state automata (Kaplan and Kay, 1994). Often rule-based G2P systems also incorporate a dictionary as an exception list. While rule-based systems provide good (or even complete) coverage they have two drawbacks: firstly, designing the rules is hard and requires specific linguistic skills. Secondly, natural languages frequently exhibit irregularities, which need to be captured by exception rules or exception lists. The interdependence between rules can be quite complex, so rule designers have to cross-check if the outcome of applying the rules is correct in all cases. This makes development and maintenance of rule systems very tedious in practice. Moreover, a rule-based G2P system is still likely to make mistakes when presented with an exceptional word, not considered by the rule designer.

In contrast to the knowledge-based approach outlined above, the *data-driven* approach to grapheme-to-phoneme conversion is based on the idea that given enough examples it should be possible to predict the pronunciation of unseen words purely by analogy. The benefit of the data-driven approach is that it trades the intellectually challenging task of designing pronunciation rules, for the much simpler one of providing example pronunciations. For native speakers it is much easier to judge the correctness of a pronunciation or to write down the pronunciation of a specific word, than to formulate general spelling rules. The crucial question in data-driven G2P is how analogy should be implemented algorithmically. Starting with the work of Sejnowski and Rosenberg (1987), various machine learning techniques have been applied to this problem in the past. Before we try to give an overview in the following, we note that there are two partly competing goals in data-driven G2P, namely lexicon compression and generalization. Lexicon compression aims to minimize the storage (and computational) requirements by minimizing the error on *seen* data using a compact model. Generalization aims to overcome the limited coverage of a given dictionary by minimizing error on *unseen* data.

It is worth noting that the pronunciations used to train a data-driven G2P model ought to exemplify the pronunciation rules of the language. This is contrary to the exception list used by rule-based systems which only need to cover the atypical pronunciations. Training a model using only words with exceptional pronunciations would clearly defy any analogy-based approach. In practice, available pronunciation dictionaries which typically cover the most frequent words of the language are often used to train data-driven G2P models. While such dictionaries usually do contain atypical words, the patterns found in the more frequent, exemplary words will normally prevail. In fact, the data-driven approach mitigates the distinction between rules and exceptions. Ultimately, training data should be representative of the application domain.

2.1. Techniques based on local classification

A large group of G2P methods presuppose an alignment of the training data between letters and phonemes or create such an alignment in a separate pre-processing step. The alignment is typically construed so that each alignment item comprises exactly one letter. The number of corresponding phonemes can be zero (epsilon, or "null phoneme"), one, or greater than one, as in the following example. We call this type of alignment *1-to-n*.



Alignments can be created using hand-crafted rules, by (dynamic programming) search using predefined alignment constraints or costs, or by an iterative estimation of alignment probabilities in the spirit of the approach described in Section 3.2. In any case, the alignment problem is in this approach not part of the actual transcription method.

Typically, the input sequence is processed sequentially (e.g. from left to right). For each input character, a (possibly empty) sequence of phonemes is chosen from a small set of *allowables*. The prediction of the output phoneme (or phoneme group) is based on the context of the current letter. Since the decision for each position is taken before proceeding to the next, we call this family of techniques *local classification*. The most popular techniques used to do this prediction are neural networks and decision trees. Taking decisions about each phoneme locally is clearly not optimal from a decision theoretic point of view. How-

ever, this strategy avoids the need to use a search algorithm that is generally necessary to find the globally optimal solution.

Sejnowski and Rosenberg (1987) as well as McCulloch et al. (1987) applied neural networks to this classification problem. They use a three-layer neural network. The input of the network is a context window of plus/minus three letters. The input layer uses an orthogonal representation, i.e. one input for each type of letter. The output layer represents the predicted phoneme by means of articulatory features. Jensen and Riis (2000) and Häkkinen et al. (2003) have improved this approach by using a more sophisticated letter code-book representations in the input layer.

Torkkola (1993) uses a technique called dynamically expanding context which generates a decision tree that takes an asymmetrical window around the current letter into account. Daelemans and van den Bosch (1996) propose the use of decision trees trained using the information gain criterion (IG-Tree). Questions are used only about the surrounding letters and the information gain is computed only once for each attribute. Andersen et al. (1996) grow binary decision trees using the Gini criterion. They allow questions about letters five positions to the left and to the right of the current letter. In addition to questions for individual letters, tests for membership in 10 graphemic classes are allowed. Pagel et al. (1998) also grow decision trees using the information gain criterion but recompute the information gain for each node split. In addition to the three preceding and following letters, they allow the algorithm also to take the three following phonemes into account. This requires the word to be processed in reverse order from right to left, since the phonemes are considered to be the result of decisions taken previously. They also report improvements from adding questions about the part-of-speech (POS) of the word considered. Suontausta and Häkkinen (2000) and Häkkinen et al. (2003) also employ information gain derived decision trees. The set of possible questions includes up to four preceding and four following letters, the preceding phonemes and their phonemic classes.

2.2. Pronunciation by analogy

The term pronunciation by analogy (PbA) would be appropriate for all data-driven grapheme-to-phoneme conversion techniques, but typically it is used more specifically for approaches that could be described as nearest-neighbor-like. The common theme among PbA techniques is that they scan the training lexicon for words or parts of words that are in some sense similar to the word to be transcribed. The output pronunciation is then chosen to be analogous (in some sense) to these retrieved examples. By considering each word as a whole, PbA goes beyond local classification, but is generally not founded on a probabilistic model.

The method proposed by Dedina and Nusbaum (1991) examines every word in the lexicon and builds a pronunciation lattice structure using the phonetic representations of the words that match the input string. In this pronunciation lattice, each node represents a candidate phoneme, and each path through the lattice represents a possible pronunciation. Marchand and Damper (2000) extend and improve this approach by combining different path scoring strategies. Yvon (1996) constructs the lattice representing all potential pronunciations of a word by extracting overlapping chunks from words in the training lexicon. The transcription is obtained by determining the best path through the lattice based on maximum chunk overlap and chunk frequency.

The method described by Bagshaw (1998) operates with a set of hand-specified grapheme phoneme correspondences (GPC, cf. Section 3.2) and induces context dependent rules over these units (with a context size of two GPC positions in both directions). The final transcription is obtained by a global search over the lattice of competing segmentations with scores based on rule weights and rule violation penalties.

Bellegarda (2005) uses latent semantic analysis to define a global similarity measure for words. To transcribe an unknown word, first a set of similar lexicon entries is compiled, then all sequences in this list are aligned and for each aligned position the most frequent phoneme is chosen.

2.3. Probabilistic approaches

A number of authors have approached the G2P problem from a probabilistic perspective. Lucassen and Mercer (1984) create 1-to-n alignments of the training data using a context independent channel model. The prediction of the next phoneme is based on a symmetric window of letters and left-sided window of phonemes. To this end they induce binary feature functions using a mutual information criterion and then construct a regression tree. The leafs of this tree carry probability distributions over the set of phonemes. Jiang et al. (1997) presented an improved regression tree approach, using a refined entropy weighting scheme, smoothing of leaf distributions, bagging and rescoring with a phoneme trigram. One of the two models studied by Chen (2003) uses a similar set of feature functions but uses a conditional maximum entropy model for predicting phonemes.

Meng et al. (1994) model word pronunciations by morphological parse trees using a layered bigram as the statistical parsing approach. Besling (1994) obtains 1-to-n alignments by dynamic programming using a predefined and uniform distribution. He uses Bayes' formula to decompose the probability of candidate pronunciations into a phonotactic model, which in this case is a 7-gram on phonemes, and a matching model, which is the conditional probability of the current letter given the current phoneme as well as the previous letter and phoneme.

Some authors have proposed joint-sequence models. These models, which are central to this article, are discussed in the following section.

3. Joint-sequence models

3.1. Statistical problem formulation

An orthographic form is given as a sequence of letters, also referred to as characters or graphemes. We denote the set of graphemes as G. A pronunciation is represented in terms of a phonemic transcription, i.e. a sequence of phoneme symbols. The set of phonemes is denoted as Φ. The task of grapheme-to-phoneme conversion, or phonetic transcription, can be formalized using Bayes' decision rule as

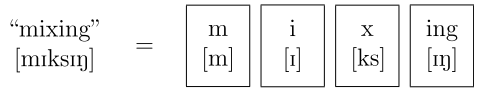
$$\boldsymbol{\varphi}(\boldsymbol{g}) = \operatorname*{argmax}_{\boldsymbol{\varphi}' \in \Phi^*} p(\boldsymbol{g}, \boldsymbol{\varphi}') \tag{1}$$

This means, for a given orthographic form $\boldsymbol{g} \in G^*$ we seek the most likely pronunciation $\boldsymbol{\varphi} \in \Phi^*$. Here V^* denotes the set of all strings over symbols in V (Kleene star). It should be noted that this decision strategy is optimal with respect to word error, i.e. the risk of not getting the correct pronunciation is minimized.

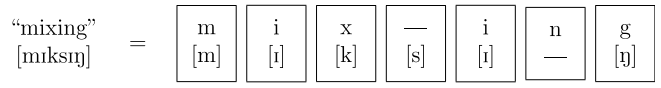
3.2. Co-segmentations and graphones

The fundamental idea of joint-sequence models is that the relation of input and output sequences can be generated from a common sequence of joint units which carry both input and output symbols. In the simplest case, each unit carries zero or one input and zero or one output symbol. This corresponds to the conventional definition of finite state transducers (FST). When units may carry multiple input and output symbols, the terms *co-sequence* and *joint multigram* (Deligne et al., 1995) are used. While the approach is applicable to any monotonous translation problem, it is formulated here in the context of grapheme-to-phoneme conversion. As in our previous study (Bisani and Ney, 2002) we refer to joint units here as *graphones*. Other terms used to refer to joint units in the context of grapheme-to-phoneme conversion are *grapheme-to-phoneme correspondences* (GPC) (Galescu et al., 2001) and *graphonemes* (Vozila et al., 2003).

A grapheme–phoneme joint multigram, or *graphone* for short, is a pair $q = (\boldsymbol{g}, \boldsymbol{\varphi}) \in Q \subseteq G^* \times \Phi^*$ of a letter sequence and a phoneme sequence of possibly different length. We use the expressions $\boldsymbol{g}_q$ and $\boldsymbol{\varphi}_q$ to refer to the first and second component of q, respectively. We call a graphone *singular*, if it has at most one letter and at most one phoneme. The inventory of graphones Q can be inferred automatically from training data (cf. Section 4) or it can be specified by hand. In the joint multigram model it is assumed that for each word its orthographic form and its pronunciation are generated by a common sequence of graphones. For example, the pronunciation of "mixing" may be regarded as a sequence of four graphones:



The letter and the phoneme sequences are grouped into an equal number of segments. Such a grouping is called a joint segmentation, or *co-segmentation*. The more general term alignment is often used interchangeably. We refer to this particular type of alignment as *m-to-n*. For a given pair of input and output strings, the segmentation into graphones may not be unique. Compared to the methods described in Section 2.1, which may also have alignment ambiguity, *m*-to-*n* alignments have the additional freedom of how input letters are grouped. For example the following segmentation into seven singular graphones is equally valid. We refer to this as an *01-to-01*, or FST-type alignment.



Because of this ambiguity the joint probability $p(\boldsymbol{g}, \boldsymbol{\varphi})$ is determined by summing over all matching graphone sequences:

$$p(\boldsymbol{g}, \boldsymbol{\varphi}) = \sum_{\boldsymbol{q} \in S(\boldsymbol{g}, \boldsymbol{\varphi})} p(\boldsymbol{q}) \tag{2}$$

where $\boldsymbol{q} \in Q^*$ is a sequence of graphones and $S(\boldsymbol{g}, \boldsymbol{\varphi})$ is the set of all co-segmentations of $\boldsymbol{g}$ and $\boldsymbol{\varphi}$:

$$S(\boldsymbol{g}, \boldsymbol{\varphi}) := \left\{ \boldsymbol{q} \in Q^* \,\middle|\, \begin{matrix} \boldsymbol{g}_{q_1} \smile \cdots \smile \boldsymbol{g}_{q_K} = \boldsymbol{g} \\ \boldsymbol{\varphi}_{q_1} \smile \cdots \smile \boldsymbol{\varphi}_{q_K} = \boldsymbol{\varphi} \end{matrix} \right\} \tag{3}$$

Here $\smile$ denotes sequence concatenation and $K = |\boldsymbol{q}|$ is the length of the graphone sequence $\boldsymbol{q}$. The joint probability distribution $p(\boldsymbol{g}, \boldsymbol{\varphi})$ has thus been reduced to a probability distribution $p(\boldsymbol{q})$ over graphone sequences $\boldsymbol{q} = q_1, \ldots, q_K$, which can be modeled using a standard M-gram approximation:

$$p(q_1^K) \cong \prod_{j=1}^{K+1} p(q_j | q_{j-1}, \ldots, q_{j-M+1}) \tag{4}$$

Positions $i < 1$ and $i > K$ are understood to hold a special boundary symbol $q_i = \perp$ which allows modeling of characteristic phenomena at word starts and ends. In the following sections we present a novel estimation method for this type of model and continue with an experimental assessment.

3.3. Related work

Deligne et al. (1995) introduced the maximum likelihood procedure for inferring many-to-many alignments using the EM algorithm. They study two models for grapheme-to-phoneme conversion based on this. One uses a joint unigram model on multigrams, the other uses a Bayes decomposition in to a phonotactic bigram and a context independent matching model. In a previous study the multigram approach was combined with a joint trigram model (Bisani and Ney, 2002). Galescu and Allen (2002)

and Vozila et al. (2003) use a very similar approach with a joint 4-gram model and a different alignment method.

In contrast to these "chunk"-based methods, it is also possible to build a joint model with only singular graphones. Caseiro et al. (2002) use FST-type alignments which they derive by a minimum edit cost criterion with manually specified costs. A joint 8-gram model is built and converted to an FST. Chen (2003) reports very good results with a structurally similar model. He uses a maximum entropy 8-gram with Gaussian priors and obtains alignments by EM training of the model.

4. Model estimation

4.1. Multigram inference through expectation maximization

In the following, we consider the problem of inferring a model using variable length units from training data that is not co-segmented. Given is a training sample of N words paired with their pronunciations $\mathscr{O}_1, \ldots, \mathscr{O}_N = (\boldsymbol{g}_1, \boldsymbol{\varphi}_1), \ldots, (\boldsymbol{g}_N, \boldsymbol{\varphi}_N)$, but without an alignment on the level of letters and phonemes. First we note that if we have a joint-sequence model we can compute the probability of any co-segmentation for each sample, since a co-segmentation $\mathscr{S}$ uniquely defines a joint sequence:

$$p(\boldsymbol{g}, \boldsymbol{\varphi}, \mathscr{S}) = p(\boldsymbol{q}) \tag{5}$$

Thus, the log-likelihood of the training data can be formulated as the sum over all segmentations:

$$\begin{aligned}\log \mathscr{L}(\mathscr{O}_1, \ldots, \mathscr{O}_N) &= \sum_{i=1}^{N} \log \mathscr{L}(\mathscr{O}_i) \\ &= \sum_{i=1}^{N} \log \left(\sum_{\mathscr{S} \in S(\mathscr{O}_i)} p(\mathscr{O}_i, \mathscr{S}) \right)\end{aligned} \tag{6}$$

The segmentation $\mathscr{S}$ into joint units is a hidden variable. As first demonstrated by Deligne and Bimbot (1995), maximum likelihood training of this model can be performed using the expectation maximization (EM) algorithm. We first consider the context independent unigram case ($M = 1$). The re-estimation equations for the updated parameters ϑ' are

$$p(\boldsymbol{q}; \vartheta) = \prod_{j=1}^{|\boldsymbol{q}|} p(q_j; \vartheta) \tag{7}$$

$$e(q; \vartheta) := \sum_{i=1}^{N} \sum_{\boldsymbol{q} \in S(\boldsymbol{g}_i, \boldsymbol{\varphi}_i)} p(\boldsymbol{q} | \boldsymbol{g}_i, \boldsymbol{\varphi}_i; \vartheta) n_q(\boldsymbol{q}) \tag{8}$$

$$= \sum_{i=1}^{N} \sum_{\boldsymbol{q} \in S(\boldsymbol{g}_i, \boldsymbol{\varphi}_i)} \frac{p(\boldsymbol{q}; \vartheta)}{\sum_{\boldsymbol{q}' \in S(\boldsymbol{g}_i, \boldsymbol{\varphi}_i)} p(\boldsymbol{q}'; \vartheta)} n_q(\boldsymbol{q})$$

$$p(q; \vartheta') = \frac{e(q; \vartheta)}{\sum_{q'} e(q'; \vartheta)} \tag{9}$$

where $n_q(\boldsymbol{q})$ is number of occurrences of the graphone q in the sequence $\boldsymbol{q}$. The quantity $e(q; \vartheta)$, which we call the *evidence* for q, is the expected number of occurrences of the graphone q in the training sample under the current set of parameters ϑ. The evidence can be calculated efficiently by a forward–backward procedure (Deligne and Bimbot, 1997). This is further described in Section 6.1.

For higher order models ($M > 1$), we introduce the symbol h do denote the sequence of preceding joint units $h_j = (q_{j-M+1}, \ldots, q_{j-1})$. We define $n_{q,h}(\boldsymbol{q})$ to denote the number of occurrences of the M-gram $q_{j-M+1}, \ldots, q_j$ in $\boldsymbol{q}$. With this shorthand we can state the re-estimation equations as follows:

$$p(\boldsymbol{q}; \vartheta) = \prod_{j=1}^{|\boldsymbol{q}|} p(q_j | h_j; \vartheta) \tag{10}$$

$$\begin{aligned}e(q, h; \vartheta) &:= \sum_{i=1}^{N} \sum_{\boldsymbol{q} \in S(\boldsymbol{g}_i, \boldsymbol{\varphi}_i)} p(\boldsymbol{q} | \boldsymbol{g}_i, \boldsymbol{\varphi}_i; \vartheta) n_{q,h}(\boldsymbol{q}) \\ &= \sum_{i=1}^{N} \sum_{\boldsymbol{q} \in S(\boldsymbol{g}_i, \boldsymbol{\varphi}_i)} \frac{p(\boldsymbol{q}; \vartheta)}{\sum_{\boldsymbol{q}' \in S(\boldsymbol{g}_i, \boldsymbol{\varphi}_i)} p(\boldsymbol{q}'; \vartheta)} n_{q,h}(\boldsymbol{q})\end{aligned} \tag{11}$$

$$p(q | h; \vartheta') = \frac{e(q, h; \vartheta)}{\sum_{q'} e(q', h; \vartheta)} \tag{12}$$

Again sequences $\boldsymbol{q}$ are implicitly understood to start and end with a boundary symbol.

Obviously, the above equations do not permit a new graphone to emerge once its probability is zero. Therefore, we initialize the model parameters by assigning a uniform distribution over all graphones satisfying certain manually set length constraints. We typically use only a simple upper limit L, i.e. $|\boldsymbol{g}_q| \leqslant L$ and $|\boldsymbol{\varphi}_q| \leqslant L$, but exclude the non-productive case $|\boldsymbol{g}_q| = |\boldsymbol{\varphi}_q| = 0$. More complex constraints are conceivable, e.g. different limits for letter and phoneme sequence lengths, or a lower limit in addition to the upper one. The uniform initial distribution is the inverse of the total number of allowed graphones:

$$p_0(q) = \left[\sum_{l=0}^{L} \sum_{r=0}^{L} |G|^l |\Phi|^r \right]^{-1} \tag{13}$$

The summand for $r = l = 0$ accounts for the additional end-of-sequence token.

The graphone length constraint parameter L has a significant effect on the size of the resulting graphone inventory. The other external parameter of the sequence model is maximum the history length M. Together with L it defines the effective span of the model, i.e. the number of letters or phonemes that affect the estimated probabilities at a given position.

It is generally known that maximum likelihood estimates like (12) tend to over-fit the training data and predict unseen data poorly. Also with the flat initialization any graphone that can be construed from the training examples will receive some probability mass, whereas only a small subset of these is expected to contribute to the "correct" model. These two issues will be addressed by smoothing and trimming, respectively, as discussed in the following.

4.2. Evidence trimming

In a previous study the use of evidence trimming to address over-fitting was suggested (Bisani and Ney, 2002). That is, to trim the evidence values below a threshold by replacing $e(q,h;\vartheta)$ in Eq. (12) with

$$\hat{e}(q,h;\vartheta) = \begin{cases} 0 & \text{if } e(q,h;\vartheta) < \tau \\ e(q,h;\vartheta) & \text{otherwise} \end{cases} \tag{14}$$

This procedure causes the unlikely graphones to gradually die out during the iteration process. Actually, there is always implicit trimming caused by the limited machine precision. Evidence trimming is superior to model trimming where a similar thresholding is applied to the probability estimates $p(q;\vartheta)$. This is because even graphones with low probabilities $p(q;\vartheta)$ can have a conditional probability $p(q|\boldsymbol{g}_i,\boldsymbol{\varphi}_i;\vartheta)$ close to one in certain words. Removing them would leave the training sample not representable by the model. Previous experiments have shown that evidence trimming as such is effective in controlling the size of the graphone inventory (Bisani and Ney, 2002). The threshold τ needs to be adjusted on development data. In said previous study, smoothing of the M-gram model is done independently from trimming. The following section proposes an integrated smoothing technique.

4.3. Discounted evidence

Comparing the estimation equation (12) to typical n-gram language modeling, we note that we are faced with essentially the same modeling problem, except that the evidence values take the place of classical n-gram counts. It is well known that effective smoothing techniques are crucial to building good language models. Empirical studies have shown that absolute discounting with interpolation and a marginal preserving back-off distribution, also known as Kneser–Ney[1] smoothing, yields very good results and often surpasses all other known smoothing methods (Kneser and Ney, 1995; Chen and Goodman, 1999). Unlike counts in classical language modeling, evidence values are generally fractional. So care must be taken when adopting results from classical language modeling, as their derivation may rely on the assumption of integer counts. The estimation equation with absolute discounting and interpolation is

$$p_M(q|h) = \frac{\max\{e(q,h) - d_M, 0\}}{\sum_{q'} e(q',h)} + \lambda(h)\, p_{M-1}(q|\bar{h}) \tag{15}$$

For clarity we have added a subscript M to indicate the order of the distribution. $d_M \geqslant 0$ is a discount parameter. $p_{M-1}(q|\bar{h})$ is the generalized, lower order $(M-1)$-gram distribution conditioned on the reduced history $\bar{h}_i = (q_{i-M+2}, \ldots, q_{i-1})$. $\lambda(h)$ is chosen to make the overall distribution sum to one.

[1] We have chosen to adhere to common usage, with apologies by the second author for this breach of modesty.

Whereas in language modeling the smallest count value is one (apart from unseen events), evidence values can become arbitrarily small, in fact smaller than the discount. So the discounted evidence estimation includes a form of evidence trimming: graphones with evidence values below the discount parameter are excluded from the model. A notable difference between this form of evidence trimming and the explicit form (14) is that in discounting we distribute the discounted evidence over unseen events, whereas in (14) the remaining evidence is effectively distributed over seen events.

We still need to specify the back-off distribution p_{M-1}. In classical language modeling two flavors of Kneser–Ney smoothing are known: one is based on the idea of preserving the marginal distribution, the other one on leaving-one-out (Kneser and Ney, 1995; Ney et al., 1997). Because we deal with fractional "counts" it is not obvious how to apply leaving-one-out. Therefore, we follow the marginal preserving approach. The idea is to impose a consistency constraint for all reduced histories $\bar{h}$:

$$\sum_{h\in\bar{h}} p_M(q|h) \sum_{q'} e(q',h) = \sum_{h\in\bar{h}} e(q,h) \tag{16}$$

Substituting with (15) and solving for $p_{M-1}(q|\bar{h})$ under the constraint that p_{M-1} is normalized yields

$$p_{M-1}(q|\bar{h}) = \frac{\hat{e}(q,\bar{h})}{\sum_{q'} \hat{e}(q',\bar{h})} \tag{17}$$

with $\hat{e}$ being the reduced evidence

$$\hat{e}(q,\bar{h}) := \sum_{h\in\bar{h}} \min\{e(q,h), d_M\} \tag{18}$$

Of course, $p_{M-1}(q|\bar{h})$ as in (17) needs to be smoothed as well. Two approaches to smoothing p_{M-1} seem reasonable. The first is to "plug in" the reduced evidence values (18) in (15). The second is to smooth the constraints (16). It turns out that both approaches lead to equivalent results, except for different interpretation of the discount parameters. Absolute discounting applies recursively to lower order distributions $p_{M-2}, p_{M-3}, \ldots p_0$. The zerogram distribution p_0 is uniform over all potential graphones (13).

In total we have introduced M discount parameters $d_1, \ldots, d_M$. In classical language modeling, Ney et al. (1995) as well as Chen and Goodman (1999) have recommended estimates of the optimal discounts based on counts of counts. Since fractional evidence values do not lend themselves to counting, we resort to optimizing d on a held-out set using Powell's method (Press et al., 1992).

4.4. Bottom-up model construction and discounted expectation maximization

To start the interative procedure we initialize the unigram model with a flat probability distribution (13), i.e. all possible multigrams have the same initial probability. An alternative initialization uses unconstrained occurrence counts $c(q)$ in the training set (Deligne et al., 1995), that is,

counting how often a graphone potentially occurs in each word regardless of overlap with neighboring graphones.

$$c(q) := \sum_{i=1}^{N} \sum_{l_1=1}^{|g_i|} \sum_{l_2=l_1}^{|g_i|} \sum_{r_1=1}^{|\varphi_i|} \sum_{r_2=r_1}^{|\varphi_i|} \times \delta((g_{l_1} \smile \cdots \smile g_{l_2}, \varphi_{r_1} \smile \cdots \smile \varphi_{r_2}) = q) \quad (19)$$

These counts (subject to graphone length constraints) are then used to compute the initial probability distribution by applying the normal smoothing (15). Higher order M-gram models are initialized using the previously generated $(M-1)$-gram model. This means that we only allow histories which correspond to M-grams that were not discounted away in the lower order model.

We now address the question of how evidence discounting interacts with the EM algorithm. First we note that we need a data set for optimizing the discount values that is separate form the data used to compute the evidence values. Not separating these sets would lead to a gross underestimation of the discount. For this purpose we split the training data $\mathcal{O}$ into a training set $\mathcal{O}_t$ and a typically smaller held-out set $\mathcal{O}_h$. The training set is used to compute the evidence values, while the held-out set is used to adjust the discount parameters.

The normal EM algorithm strictly improves the likelihood of the training data in each iteration. This will typically lead to over-fitting and the likelihood of the held-out data will start to decrease at some point. Therefore, in the discounted EM algorithm the discount values are updated to ensure that the likelihood of the held-out data does not drop. The overall training procedure is outlined in Fig. 1.

4.5. Maximum approximation

Earlier studies of the joint multigram approach (Deligne et al., 1995) used the maximum approximation to (9) during training (the so-called Viterbi training).

$$e(q;\vartheta) \cong \sum_{i=1}^{N} n_q(\hat{\boldsymbol{q}}_i) \quad (20)$$

$$\hat{\boldsymbol{q}}_i := \operatorname*{argmax}_{\boldsymbol{q}\in\mathcal{Q}^*} p(\boldsymbol{q}|\boldsymbol{g}_i, \boldsymbol{\varphi}_i; \vartheta) \quad (21)$$

$$= \operatorname*{argmax}_{\boldsymbol{q}\in S(\boldsymbol{g}_i,\boldsymbol{\varphi}_i)} p(\boldsymbol{q};\vartheta)$$

This algorithm searches for the most likely segmentation in each iteration and derives the updated model from the graphone counts in this segmentation. Smoothing and bottom-up construction apply to this method just as described before. The validity of this approximation will be studied experimentally in Section 7.7.

5. Transcription

Having estimated the model, (1) can be applied to phonemically transcribe unseen words. In producing the phonemic transcription from the orthographic form, we usually restrict ourselves to approximating the sum in (2) by the maximum

$$p(\boldsymbol{g},\boldsymbol{\varphi}) \approx \max_{\boldsymbol{q}\in S(\boldsymbol{g},\boldsymbol{\varphi})} p(\boldsymbol{q}) \quad (22)$$

This means, we look for the most likely graphone sequence matching the given spelling and project it onto the phonemes,

$$\boldsymbol{\varphi}(\boldsymbol{g}) = \boldsymbol{\varphi}\Big(\operatorname*{argmax}_{\boldsymbol{q}\in\mathcal{Q}^*|\boldsymbol{g}(\boldsymbol{q})=\boldsymbol{g}} p(\boldsymbol{q})\Big) \quad (23)$$

The loss in accuracy incurred by this approximation will be studied empirically in Section 7.8.

It is worth noting that because of the finite history length the joint sequence model is in fact a weighted regular relation and can thus be represented by a finite state transducer. This is particularly obvious in the $L = 1$ case, when only singular graphones are permitted and each graphone can be presented by an FST transition. Each history then corresponds to a state of the FST. Of course, models with larger graphones ($L > 1$) can also be represented as an FST by introducing auxiliary states.

6. Implementation aspects

This section highlights some important aspects of a concrete implementation of the methods described thus far. Our software implementation of these techniques is available under an Open Source license at http://www-i6.informatik.rwth-aachen.de/web/Software/g2p.html.

```
for M = 1 to M_max:
    initialize M-gram model with (M-1)-gram model
        p_M(q|h) = p_{M-1}(q|h̄)
    initialize the additional discount parameter
        d_M = d_{M-1}
    repeat until L(O_h) stops increasing:
        compute evidence according to (11)
        if L(O_h) did not increase:
            adjust dicount parameters d_1, ..., d_{M-1} by direction set method
                d = argmax_{d'} L(O_h; d')
        update model according to (15) and (18)
```

Fig. 1. Discounted EM algorithm.

6.1. Training

A hash map is used to map joint-multigrams to integer indices. When long multigrams are allowed this map can consume a large amount of memory since it encompasses all possible multigrams, not only those that receive an evidence above the discount threshold. M-gram models are stored in a inverse prefix tree structure, where each node corresponds to a particular M-gram history, the root corresponding to the unigram, the leaves to the most specific histories. The tree is embedded in an array in breath-first order. So a single integer index can be used to refer to a particular M-gram history.

Evidence values (11) are computed in three steps. First segmentation graphs are constructed. Then edge posterior probabilities are computed by means of the forward–backward algorithm. Finally, evidence values are accumulated in a hash map. In the first step the set of potential co-segmentations is explicitly represented as a directed acylic graph. Each edge corresponds to a graphone q. Each vertex v of the graph corresponds to a position in the source sequence l, a position in the target sequence r and an M-gram history $h : v = (l, r, h)$. In the unigram case $(M = 1)$ there is no dependency on h and the graph can be envisioned as a rectangular grid. When the M-gram range exceeds the length of the sequence considered, the graph degenerates to a tree structure. With the explicit graph representation the computation of forward, backward and posterior probabilities can be implemented in a straight forward way. An implicit representation, i.e., embedding the graph topology in the algorithms, would greatly increase code complexity, especially for long-range M-gram models. The graph construction algorithm is based on a depth-first search, exploring the space of segmentations starting from the beginning of both sequences. Under the depth-first scheme it was possible to integrate topological sorting of the vertices into the construction algorithm, as well as removal of dead ends. The list of vertices in topological order is needed for the forward/backward computation which constitutes the second step of evidence computation.

The graphs for computing evidence values are deleted immediately after accumulation has taken place, and are re-generated in the next iteration. Keeping all graphs in memory is infeasible for larger data sets. The same graph construction is used to compute the likelihood of the held-out set. Since the held-out set is typically small, the graphs are not deleted for improved efficiency, especially during adjustment of discount parameters. To provide a sense of proportion: training of the 9-gram model on the Pronlex data set, which is one of the larger data sets we have used, took about 3 days on a 1.8 GHz CPU and required up to 1GB of RAM.

6.2. First-best search

The optimization problem (23) can be viewed as a graph search problem and can be solved using dynamic programming techniques. The open choice is which search strategy is to be employed. An obvious choice would be to work synchronously on the input side, i.e. considering each input letter after the other. A potential problem is that when allowing graphones with an empty grapheme side (i.e. input epsilon) in principle an infinite number of graphones can be concatenated before advancing to the next input position. In practice very large potential search spaces can be handled by heuristically pruning the set of partial hypotheses. This technique is known as beam search. It has the risk of search errors, when the beam width is too tight. The beam width is a tunable parameter controlling the run-time vs. accuracy trade-off. Chen (2003) uses beam search, whereas Galescu and Allen (2002) do not allow null grapheme units and use simple dynamic programming search. We have chosen to implement a best-first search strategy, where the current partial hypothesis with the highest probability is expanded first. Alternative paths to the same node are recombined, that is when a node is reached a second time via an alternate path only the best scoring alternative is pursued further. This procedure is equivalent to Dijkstra's algorithm, or the A*-algorithm with zero rest cost. This algorithm is exact (no search errors), and we have found it to be highly efficient for this problem. The computational effort on a typical PC was so small that further investigations into optimizing the search strategy were not considered worth while.

6.3. N-best search and posterior probabilities

A second decoding algorithm was implemented which is computationally more expensive, but also provides additional results: first, a best-first search is conducted as described above, but instead of relaxing alternative paths to nodes that have already been seen, they are stored in a graph structure. The resulting graph encodes all possible translations of the source sequences. Each path uniquely corresponds to a co-segmentation of the source sequence and a possible translation. Executing the forward algorithm with summation on the graph, yields the probability of the source sequence very efficiently

$$p(\boldsymbol{g}) = \sum_{\boldsymbol{\varphi} \in \Phi^*} p(\boldsymbol{g}, \boldsymbol{\varphi}) = \sum_{\boldsymbol{q} \in Q^* | \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{g}} p(\boldsymbol{q}) \tag{24}$$

This makes it easy to compute the posterior probability of a translation:

$$p(\boldsymbol{\varphi} | \boldsymbol{g}) = \frac{\sum_{\boldsymbol{q} \in S(\boldsymbol{g}, \boldsymbol{\varphi})} p(\boldsymbol{q})}{p(\boldsymbol{g})} \tag{25}$$

After the graph has been constructed, another A*-search is conducted on the graph to generate n-best translations. This second search works in reverse direction starting from the final node of the graph. The forward probabilities without summation are used as a perfect rest cost. Therefore, the paths through the graph are retrieved in the right order from highest to lowest probability. N-best list generation is a theoretically pleasing way of generating alternative pronunciation candidates. We will comment on this possibility

in Section 9.1. The value of the posterior probability as a confidence measure is examined in Section 9.4.

In practice we find that it is not necessary to carry out the summation in the numerator of Eq. (25). Often no alternative segmentation for the best translation occurs among the top-scoring candidates (cf. Section 7.8). Therefore, we will use the following approximation in most experiments:

$$p(\boldsymbol{\varphi}|\boldsymbol{g}) \approx \frac{\max_{\boldsymbol{q}\in S(\boldsymbol{g},\boldsymbol{\varphi})} p(\boldsymbol{q})}{p(\boldsymbol{g})} \tag{26}$$

7. Experiments

7.1. Performance metrics

Accuracy of grapheme-to-phoneme conversion is measured in terms of *phoneme error rate* (PER), which is the edit distance between the automatic transcription result (candidate) and reference pronunciation divided by the number of phonemes in the reference pronunciation. Edit distance (or Levenshtein distance) is the minimum number of insert, delete and substitute operations required to transform one sequence into the other (Levenshtein, 1965). If the reference lexicon contains multiple pronunciation variants for a word, the variant that has the smallest edit distance to the candidate is used. An additional performance metric is *word error rate* (WER), which is the relative proportion of words that have at least one phoneme error. It is less forgiving to near misses than phoneme error rate.

Several other publications report phoneme and word accuracy instead of error rates. For words the relation is very simple, word error rate is one minus word accuracy. Phoneme accuracy is frequently used in studies using the letter classification paradigm (cf. Section 2.1). Unfortunately, direct translation to phoneme error rate is not always possible, because it is often not clear whether accuracy is normalized on the number of (correct) phonemes or on the number of letters, or how the case of multiple output phonemes for one letter ("pseudo phoneme") is counted. We therefore refrain from converting phoneme accuracies given in other publications.

7.2. Data sets

In order to find out how the proposed method compares in terms of accuracy to methods published previously, we have tested it on a variety of English pronunciation databases. Table 1 gives an overview of the data sets used. In all cases the original database is partitioned randomly into disjoint training and testing sets. We have tried to replicate the conditions reported in previous studies, so that the performance figures can be compared directly.[2] Two of the data sets were made available as part of the Pascal Letter-to-Phoneme Conversion Challenge (van den Bosch et al., 2006). For the NETtalk database (Sejnowski and Rosenberg, 1993) three different replications were used to reproduce different settings used in other publications. The approximate size of the training set is used to indicate a particular replication. The 19k variant excludes homographs and one-letters words. The two other US English databases are CMUdict (Weide, 1998) release 0.6 and Pronlex (Kingsbury et al., 1997). Three databases of British English were used: Beep (Robinson, 1997) as used in the Pascal Challenge, OALD (Mitton, 1992), and Celex (Celex, 1995). The grapheme set of the Beep database includes apostrophe, hyphen, underscore and period. (It also contains some other rare punctuation characters that were not counted in Table 1.) The OALD phoneme set includes stressed and unstressed vowels. The Celex set is a randomly chosen subset of the actual Celex database which excludes phrases, abbreviations and homographs and in which all words were converted to lower case. In addition, two non-English databases were used: the German database LexDb (Lüngen et al., 1998) and the French database Brulex (Content et al., 1990). The LexDb set is a random subset of the actual LexDb database. It excludes hyphenated compounds, abbreviations and pronunciation variants and has all words converted to lower case. The Brulex data set again comes from the Pascal Challenge.

7.3. Convergence behavior

To illustrate the behavior of the proposed training algorithm an example of the evolution of training data likelihood and error rate over training iterations is shown in Fig. 2. The held-out set likelihood increases monotonically. This is ensured by periodically adjusting the discount parameters. We observe that typically the discount slowly grows from iteration to iteration. As one may expect the discount increases with model order $d_{M-1} < d_M$.

7.4. Size of graphones

The effective range covered by the model is controlled by the length of graphones and the order M of the sequence model. The actual size of each graphone is an outcome from the training procedure. However the maximum size of graphones considered is controlled by a parameter L.

Here all graphones of zero up to L letters and phonemes are allowed. On the NETtalk 15k data set all combinations of L and M in range from 1 to 4 and 1 to 6, respectively, have been tested. The results are depicted in Fig. 3. We see that performance monotonically improves with longer M-gram range. Concerning the maximum graphone size two regimes can be distinguished: When the sequence model retains little or no context information ($M \leqslant 2$), chunks can step in and performance is better the longer the chunks are. With longer range sequence models ($M \geqslant 4$) the situation is reversed and accuracy is worse

[2] We would like to express our gratitude to Stanley F. Chen who was kind enough to share the pre-processing and data set partitioning he used in (Chen, 2003).

Table 1
Overview of pronunciation databases used in experiments on grapheme-to-phoneme conversion

	Symbols		Word length		Prons/words	Number of words			x-validation
	$\|G\|$	$\|\Phi\|$	$\|\boldsymbol{g}\|$	$\|\boldsymbol{\varphi}\|$		Train	Test	Held	
British English									
Beep	30	45	9.0	7.6	1.073	215,713	25,706	–	10×
Celex	26	53	8.4	7.1	1	39,995	15,000	5000	–
OALD	26	82	8.2	6.9	1.008	56,961	6377	–	–
US English									
NETtalk 15k	26	50	7.3	6.2	1.010	14,851	4951	–	–
NETtalk 18k	26	50	7.3	6.3	1.010	17,822	1980	–	10×
NETtalk 19k	26	50	7.3	6.3	1	18,595	1000	–	5×
CMUdict	27	39	7.5	6.3	1.062	106,837	12,000	–	–
Pronlex	30	41	7.4	6.9	1.094	83,182	4800	2400	–
German									
LexDb	30	46	10.4	9.0	1	40,000	15,000	5000	–
French									
Brulex	40	39	8.5	6.7	1	24,726	2747	–	10×

Size of grapheme and phoneme inventory, average length of words, average number of pronunciations per word in the training set, number of orthographically distinct words in training, test and held-out set. In case of cross-validation averages are given.

the bigger the chunks are. This may be expected as data are sparser for larger chunks. With four-grams and beyond the best results are obtained with using singular graphones only ($L = 1$).

An additional experiment was conducted to study 1-to-n alignments. This was done by choosing a different constraint on graphone length: all graphones have exactly one letter, and zero up to L phonemes, i.e. $|\boldsymbol{g}_q| = 1$ and $|\boldsymbol{\varphi}_q| \leqslant L$. This type of alignment emulates typical local classification approaches (cf. Section 2.1). Results are given in Table 2. We observe that for 1-to-L alignments the performance is virtually independent of L. The restriction to exactly one letter per graphone alone constrains the set of segmentations largely, so that the additional constraint on the number of phonemes does not play a significant role. For $L = 1$ accuracy is slightly higher and is on par with the FST-type model that does additionally allow phoneme insertions. Considering that most words have fewer phonemes than letters, this is understandable.

7.5. Size of held-out set

As explained above, it is necessary to reserve some part of the training data as a held-out set which is used to adjust the discount parameters. An open question is how large the held-out set should be. On the one hand, too small a held-out set will make the estimation of discounts unreliable, leading to poor performance. On the other hand, enlarging the held-out set reduces the amount of data used in the actual model estimation, which also deteriorates performance. The second problem can be partially alleviated by doing additional *fold-back* training: After the training process has converged, the held-out data is added to the estimation data and training is iterated further until convergence while keeping the discount parameters fixed.

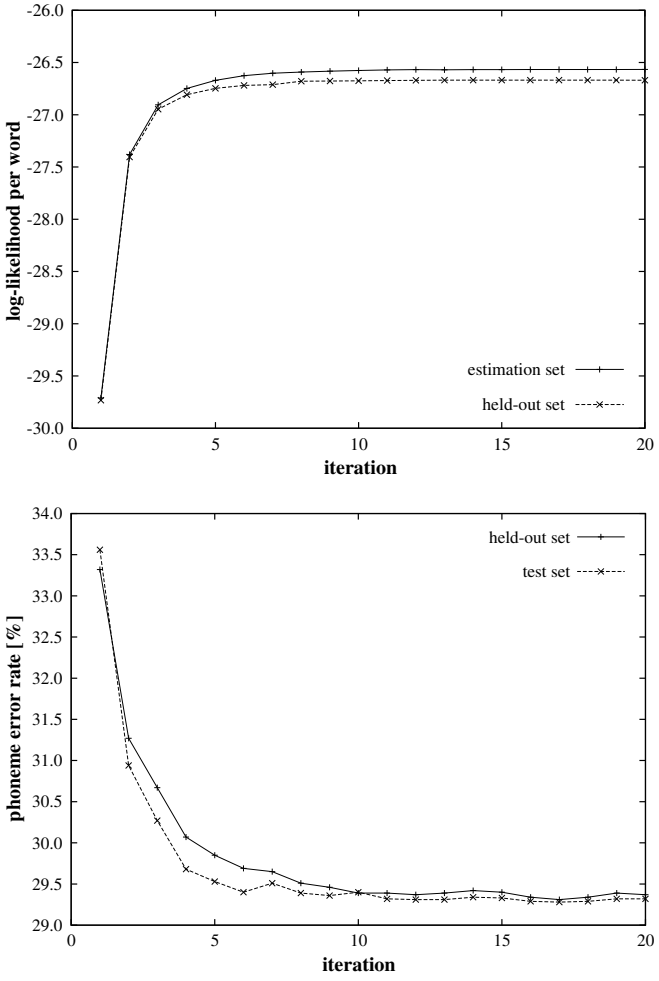


Fig. 2. Evolution of training data likelihood and model performance over EM training iterations. NETtalk 15k corpus, $L = 2,\ M = 1$.

To find the best trade-off, we have done experiments on the NETtalk 15k data set with varying held-out set sizes

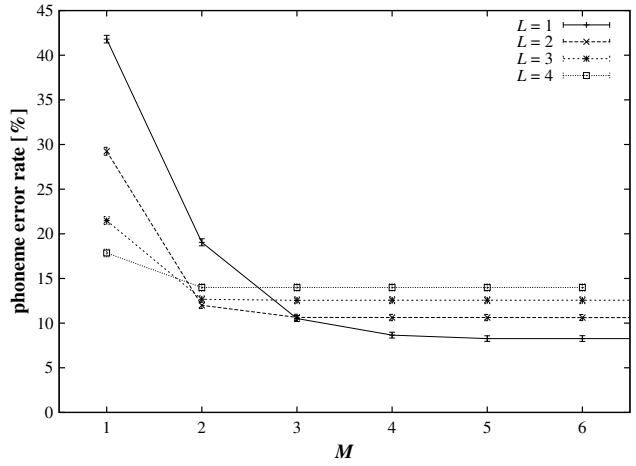


Fig. 3. Grapheme-to-phoneme conversion performance on the NETtalk 15k data set for L-to-L models of different complexities. Error bars show 90% confidence intervals.

Table 2
Comparison of different types of alignment

Alignment type	$L = 1$	$L = 2$	$L = 3$	$L = 4$
L-to-L	8.27	10.60	12.69	13.99
1-to-L	8.27	8.32	8.33	8.32

Phoneme error rates in percent on NETtalk 15k, $M = 6$.

using $L = 1$ and $M = 6$. Fig. 4 shows that there is a shallow optimum at a held-out set size of about 1000 words when fold-back is employed which is 7% of the training data for this data set. Without fold-back the depletion of training examples makes itself felt very strongly. We conclude that fold-back is generally advisable. Since the held-out data is used to estimate the M discount parameters, a held-out set of 1000 words should also work well with larger data sets. In the following experiments we have generally used fold-back, except on data sets which have a dedicated development test set.

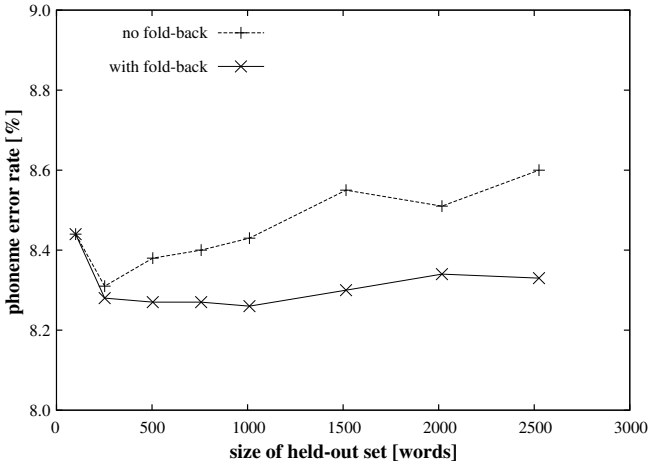


Fig. 4. Graphme-to-phoneme conversion performance on the NETtalk 15k data as a function of different partitionings of the training data ($L = 1$, $M = 6$). For the no fold-back case, the set of words used for model estimation is reduced by the respective amount of held-out data. In the fold-back case additional EM training iterations with the full training set were done.

7.6. Effect of smoothing

In the theoretical derivation of the probabilistic model much attention was given to smoothing. In this section we want to verify our assumption about the importance of smoothing empirically. In a first attempt to do this all discount parameters were fixed to be zero. This emulates maximum likelihood estimation as far as possible with our software implementation. Because we represent all probabilities in the log-domain, zero probabilities are actually taken to be approximately $10^{-10^{10}}$. For this reason, when the decoder encounters an unseen grapheme sequence it will use the back-off "distribution" even though it is penalized with a quasi-zero weight. As there is no discounting the back-off "distribution" is uniform. A true implementation of naive maximum likelihood would simply bail out when presented with an grapheme M-gram that was not seen in training. Therefore, we think that the back-off as last resort scheme is more reasonable to compare to. Nevertheless, it performs quite poorly. As shown in Fig. 5, over-fitting kicks in at $M \geqslant 4$. The lowest phoneme error rate on NETtalk 15k is 11.78% at $M = 3$ (and $L = 1$), which is 42% higher than what we achieve with the smoothed model.

The main shortcoming of the unsmoothed model is that it is completely indiscriminative on the unseen events. In a second experiment the discount was set to a small but non-zero value $d = 10^{-6}$. For comparison, the empirical discounts determined during training as described are on this data $d_1 = 0.06$, $d_2 = 0.31$, $d_3 = 0.54$, $d_4 = 0.68$, $d_5 = 0.79$. Although this model assigns too little probability mass to unseen events, when it must back-off it makes use of a proper back-off distribution. We find that this marginally smoothed model performs much better than the

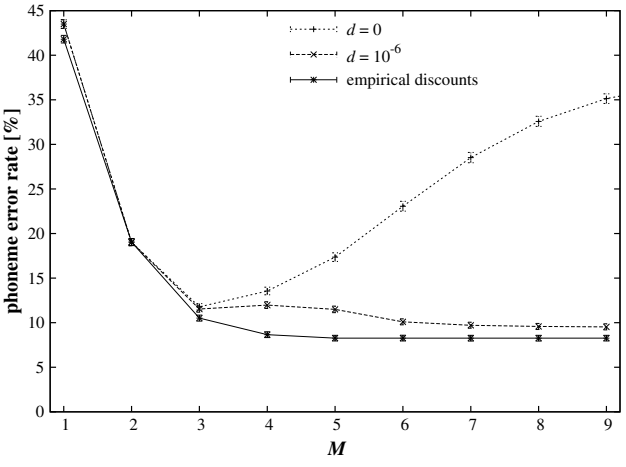


Fig. 5. Grapheme-to-phoneme conversion accuracy on NETtalk 15k for smoothed and unsmoothed models as a function of context length ($L = 1$). Error bars show 90% confidence intervals.

unsmoothed one, but still lags far behind the empirically smoothed model (see Fig. 5). For short context lengths ($M \leqslant 2$) error rates are near identical, as data is not sparse in this regime. Beyond that, the error rate of the smoothed model drops rapidly and reaches a plateau at 8.27% for $M \geqslant 5$. For the marginally smoothed model the error rate decreases more gradually and reaches its lowest value of 9.47% at $M = 11$. Thus, the smoothed models achieve a 12.7% relative lower error rate with a much smaller model (151k vs. 435k M-grams).

7.7. Training with maximum approximation

We now address the question whether the maximum approximation during training is allowable (cf. Section 4.5). Models were trained on the NETtalk 15k data set with maximum graphone sizes of $L = 1, \ldots, 3$ using four different settings. The model was initialized either flat (13), or using unconstrained counts (19). Subsequently either EM training ("sum", i.e. with summation over alternative segmentations), or training with maximum approximation ("maximum") was performed to convergence.

Results in terms of transcription accuracy are given in Table 3. We find that training with maximum approximation is very sensitive to initialization. The count-based initialization yields far better results for non-singular graphones. For singular graphones the advantage is not quite significant. Training with summation is rather insensitive to initialization. The difference between both initialization schemes is insignificant in this case. Generally the maximum approximation in training hurts performance. For $L = 2$ the difference is significant with a probability of improvement of 95% according to a pair-wise bootstrap analysis (Bisani and Ney, 2004). For $L = 1$ there seems not to be a significant disadvantage to using the maximum approximation.

7.8. Transcription with summation

The experiment described in this section aims at determining the impact of the maximum approximation (22) on transcription accuracy. Carrying out the sum over different alignments of the same target sequence can be executed with the help of n-best lists. N-best lists were computed for the NETtalk 15k corpus as described in Section 6.3 using models with $M = 6$. Each entry in the list represents a unique graphone sequence which implies the candidate transcription and has a posterior probability associated with it. The length of each n-best list was chosen to be at least 50 and for each word so that the accumulated posterior probability was greater than 0.99. As a result, the oracle error of the n-best lists is far below the first-best one. (By oracle error rate we mean the error rate that could be obtained by picking the best hypothesis from each list.) The same transcription may occur multiple times in such an n-best list, each time generated by a different alignment. We measure how often these repetitions occur by stating the ratio of the number of entries to the number of distinct pronunciations. From each original n-best list two new lists were derived. For the maximum-approximation n-best list, any repeated pronunciations were simply dropped and the list was left in its original order which is by decreasing maximum-approximated posterior probability (26). For the n-best list with summation, we summed the posterior probabilities of the entries which correspond to the same pronunciation. This list was re-sorted according to this better approximation of the true posterior probability. The average rank-order correlation was computed to quantify how much the two resulting lists differ. In addition we counted for what fraction of words the top-scoring candidate changed. We also compared the error rates for the top-scoring candidate in both cases.

The results are given in Table 4. We find that repeated transcriptions are very rare in the case of singular graphones ($L = 1$). As a consequence accuracy results are practically unaffected by carrying out the summation in decoding. For the models employing larger graphones there is more ambiguity in the alignment of source and target sequence. This manifests itself in a larger number of alignments for each candidate transcription. Still the rank-order correlation is very close to one, implying that different candidates change position only rarely due to summation. In terms of accuracy, summation in decoding leads to a small but consistent improvement. Still, the best results are obtained with singular graphones where summation is unnecessary.

7.9. Performance evaluation

The following experiments are meant to evaluate how well the method proposed here works on a variety of data

Table 3
Comparison of model initialization and training schemes

Initialization	Training	$L = 1$	$L = 2$	$L = 3$
Flat	Maximum	8.42	12.40	20.71
Counts	Maximum	8.36	10.84	13.65
Flat	Sum	8.27	10.60	12.66
Counts	Sum	8.27	10.59	12.61

Phoneme error rates in percent on NETtalk 15k with $M = 6$. Training was performed either with maximum approximation, or with normal EM ("sum").

Table 4
Comparison of grapheme-to-phoneme conversion with and without using the maximum approximation in decoding

	$L = 1$	$L = 2$	$L = 3$	$L = 4$
Length of n-best list	84.00	101.07	56.96	47.14
Distinct pronunciations	81.91	69.19	36.72	29.29
Alignments per pronunciation	1.0184	1.4330	1.6378	1.7108
Oracle PER [%]	0.23	0.46	1.43	3.74
Rank-order correlation	0.9838	0.9864	0.9796	0.9524
First-best changes [%]	0.0800	1.9992	2.9035	2.2065
PER without summation [%]	8.27	10.60	12.69	14.89
PER with summation [%]	8.27	10.42	12.43	14.75

NETtalk 15k, $M = 6$.

sets, especially in comparison with other methods. We applied the settings that were found to be best in the experiments described above. It is conceivable that different settings work better when data sets of significantly larger size are used. However, we did not pursue this possibility, as preliminary tests have indicated otherwise.

The settings used can be summarized as follows: models used only singular graphones and were trained with the EM algorithm (with summation) and flat initialization. When the test set did not have a dedicated development test set, a held-out set was created by randomly picking about 1000 words from the training data. In this case additional fold-back training was done as a final step. On data sets that have a development test set (Celex, Pronlex and LexDb) it was used as the held-out set in training, but no fold-back training was done. For each data set the sequence model order was chosen that gave the highest log-likelihood for the held-out set. In all cases this was $M = 8$ or 9. In decoding the maximum approximation was used.

Table 5 summarizes our results on all English data sets and quotes results published by other authors. Results on non-English data sets are listed in Table 6. Unfortunately, to our knowledge, no error rates have been published on these data sets by other authors. Error margins corresponding to 90% confidence intervals were computed using per-word bootstrap resampling (Bisani and Ney, 2004). When the data set prescribed the use of n-fold cross-validation, n models were trained independently on $n - 1/n$th of the data and evaluated on the remaining nth. Error rates are in this case computed on the union of the n test sets, and error margins are obtained from sampling over this union. For reference the sizes of all models in terms of their number of graphones and m-grams are reported in Table 8.

Table 5
Summary and comparison of grapheme-to-phoneme conversion accuracy on English data sets

Data set		Author	PER [%]	WER [%]
Beep		This work	3.38 ± 0.03	20.08 ± 0.15
Celex	=	Bisani and Ney (2002)	3.98	
	=	Vozila et al. (2003)	3.68	17.13
		Chen (2003)	2.7	
	=	This work	2.50 ± 0.11	11.42 ± 0.43
OALD		Pagel et al. (1998) with POS	6.03	21.87
		Pagel et al. (1998) w/o POS		23.34
	=	Chen (2003)		18.9
	=	This work	3.54 ± 0.19	17.49 ± 0.78
NETtalk 15k		Andersen et al. (1996)		47.0
		Jiang et al. (1997)	8.1	34.2
		Chen (2003)		34.6
		This work	8.26 ± 0.32	33.67 ± 1.10
NETtalk 18k		Torkkola (1993)	9.2	
		Yvon (1996)		36.04
		Galescu et al. (2001)	9.00	36.07
		This work	7.83 ± 0.16	31.79 ± 0.54
NETtalk 19k		Marchand and Damper (2000)		34.5
		Chen (2003)		32.1
		This work	7.66 ± 0.31	31.00 ± 1.09
CMUdict		Galescu and Allen (2002)	7.0	28.5
	=	Chen (2003)	5.9	24.7
	=	This work	5.88 ± 0.18	24.53 ± 0.65
Pronlex	=	Chen (2003) conditional ME	8.00	31.8
	=	Chen (2003) joint ME	7.15	27.3
	=	This work	6.78 ± 0.31	27.33 ± 1.04

The lines marked with "=" use exactly the same data both for training and testing. Other lines use faithful replications. "±" indicates 90% confidence interval.

Table 6
Grapheme-to-phoneme conversion accuracy for non-English data sets

Language	Data set	PER [%]	WER [%]
German	LexDb	0.28 ± 0.03	1.75 ± 0.18
French	Brulex	1.18 ± 0.05	6.25 ± 0.24

Table 7
Grapheme-to-phoneme conversion accuracy on the respective *training* sets

Language	Data set	PER [%]	WER [%]
British English	Beep	0.36 ± 0.01	2.18 ± 0.05
US English	NETtalk 15k	0.44 ± 0.04	2.16 ± 0.20
US English	CMUdict	0.38 ± 0.02	1.79 ± 0.07
German	LexDb	0.007 ± 0.002	0.06 ± 0.02
French	Brulex	0.05 ± 0.01	0.25 ± 0.05

The same models as in Tables 5 and 6 are used.

Table 8
Sizes of grapheme-to-phoneme conversion models corresponding to the accuracies reported in Tables 5 and 6

Data set	Graphones	Order	Parameters
Beep	485	9	1,728,389
Celex	226	8	670,582
OALD	307	9	815,109
NETtalk 15k	149	8	246,149
NETtalk 18k	158	8	293,572
NETtalk 19k	155	8	296,184
CMUdict	270	9	1,556,784
Pronlex	282	8	1,321,060
LexDb	158	9	347,758
Brulex	193	8	401,739

Total number of graphones occurring in the model (after trimming), M-gram order of the model, and total number of M-grams stored (including back-off weights). For data sets using cross-validation the maximum values are given.

8. Discussion

The numbers in Table 5 show that the proposed method is more accurate than or on par with all previously published result. Apart from that, these results as well as the various contrastive experiments reported provide some insight on previously unanswered questions. First we should point out that the results by Chen (2003) are very close to those obtained by the method presented here. This comes as no surprise as his approach is very similar, albeit

computationally more demanding. He uses only singular graphones with a long-range M-gram model, which is the same basic configuration that yielded the best performance in our experiments. However, instead of a discounting and interpolation type model estimation he uses a maximum-entropy-based model with Gaussian priors. Our results show that the maximum-entropy method is not essential for obtaining this high level of accuracy. As both models make use the same contextual information (M-grams), smooth probabilities of unseen events and preserve lower-order marginals, similar levels of performance can be expected (Chen and Rosenfeld, 2000).

An advantageous property of joint-sequence models is their capability to handle the alignment problem intrinsically (Och and Ney, 2003). This is convenient especially in developing a grapheme-to-phoneme conversion system for a new language where otherwise alignment rules would need to be written by hand. Intuitively the ability of this model to group symbols into larger units ("chunking") is appealing as it allows for a natural mapping of frequent letter groups such as "th" or "ph". Table 9 shows some examples of automatically inferred graphones. Chen (2003) doubted whether chunking was really beneficial to grapheme-to-phoneme conversion accuracy, but it should be noted that no previous work has explored the use of non-singular graphones in combination with a long-range M-gram ($M \geqslant 5$) sequence model. The results in Section 7.4 confirm previous reports that non-singular graphones help when the overarching M-gram sequence model has a short span. However, as the span of the M-gram model is extended, shorter graphones gain in accuracy more quickly than longer ones, and eventually perform better. Thus, singular graphones in combination with a long-range sequence model yields the best performance. Accuracy increases monotonically with M, and typically saturates at $M = 8$ or 9, which corresponds to typical word length and confirms the common language modeling wisdom that one should "remember" everything. It was further shown that 1-to-n alignments can achieve comparable accuracy. Such alignments may have advantages in terms of a simpler decoder implementation.

The failure of non-singular graphones can partially be explained by considering that larger graphone inventories aggravate data sparseness in estimating the sequence model. On the other hand, this shows that we have not found the best conceivable training procedure, yet. Since the shorter graphones are a subset of the larger ones, a perfect algorithm should be able to choose only the shorter ones if they predict pronunciations best, and artificial constraints should be unnecessary. Still the ability of the algorithm presented to infer variable length fragments from unaligned data may be useful in other applications, for example in finding sub-word units for open-vocabulary speech recognition (Bisani and Ney, 2005; Galescu, 2003).

Concerning the use of the maximum approximation, experiments confirm theory in that carrying out the summation consistently leads to better performance. During training it is generally advisable to use the true EM algorithm (with summation). For transcription, summation does not have an impact, as long as only singular graphones are used. We have also demonstrated that smoothing is essential to obtaining highly accurate models.

Table 9
Typical examples of automatically inferred graphones

$p(q)$	g_q	φ_q	$p(q)$	g_q	φ_q
(a) English Celex			*(b) German LexDb*		
0.04825	"s"	[s]	0.07438	"en"	[ə n]
0.03134	"t"	[t]	0.05756	"t"	[t]
0.02647	"s"	[z]	0.03535	"ge"	[g ə]
0.02446	"ing"	[ɪ ŋ]	0.03026	"n"	[n]
0.02116	"l"	[l]	0.02592	"r"	[r]
0.02067	"p"	[p]	0.02204	"l"	[l]
0.01994	"n"	[n]	0.02201	"s"	[s]
0.01845	"d"	[d]	0.02177	"te"	[t ə]
0.01817	"st"	[s t]	0.01927	"sch"	[ʃ]
0.01166	"in"	[ɪ n]	0.01907	"m"	[m]
0.01114	"m"	[m]	0.01905	"de"	[d ə]
0.01073	"ly"	[l ɪ]	0.01725	"st"	[s t]
0.00997	"b"	[b]	0.01555	"e"	[ə]
0.00981	"c"	[k]	0.01492	"es"	[ə s]
0.00813	"tion"	[ʃ n̩]	0.01418	"er"	[ɐ]

These are the 15 graphones with the highest unigram probabilities inferred from the English Celex (a) and the German LexDb (b) database. $M = 1,\ L = 4$.

By evaluating on unseen data this study has emphasized the generalization capabilities of the grapheme-to-phoneme converter. Yet, a major strength of sequence models of this type is that they easily memorize long sequences from the training data. This manifests itself in very low error rates on the training set, shown in Table 7. However, the models require a quite large amount of memory and are probably not suitable for lexicon compression. Further research is required for application in scarce memory environments.

9. Applications

9.1. Lexicon augmentation

Large vocabulary speech recognition (LVCSR) systems rely on phonemic transcriptions to build hidden Markov models for all the words in their recognition vocabulary from smaller context dependent units (e.g. triphone models). In typical application domains the recognition vocabulary will have reasonably high but rarely complete coverage. When applications allow end users to add words to the recognition vocabulary the phonemic representation is typically hidden, because the user is not expected to understand and use the phonetic notation. In this case grapheme-to-phoneme conversion is used to produce a pronunciation for the orthographic form provided by the user. Desktop applications may additionally make use of acoustic sample utterances.

Grapheme-to-phoneme conversion can also be used in rapid cross-domain porting. In this scenario one wants to

adapt an existing LVCSR system to work in a new application domain. Typically the existing lexicon will have a rather high out-of-vocabulary (OOV) rate on the new domain and a large number of words must be added to the system's dictionary. Here grapheme-to-phoneme conversion is a fast and cheap way to provide the missing phonetic transcriptions. In (Bisani and Ney, 2003) we studied the trade-off between manual transcription effort and speech recognitions accuracy. The advantage of data-driven methods such as the one presented here is that the model can be trained on the existing system dictionary. Therefore, the automatically produced transcriptions will be consistent with the pronunciations taken over from the existing dictionary. Gollan et al. (2005) as well as Lööf et al. (2006) report on systems that have been ported to a new domain using this technique. A grapheme-to-phoneme converter that was developed independently will generally deviate in terms of the phoneme inventories and transcription conventions used.

Speech recognition systems often use multiple pronunciations for a word to account for some variation in the way users speak. The *n*-best decoder described in Section 6.3 allows generation of alternative pronunciation candidates. If the training data contains (systematic) pronunciation variation, the *n*-best list decoder will consistently produce variants. However, there is little guidance as to how many pronunciations should be accepted in this case. It should also be noted that the algorithm will not come up with likely variants in the sense of natural phonological variation. The variants generated will reflect the ambiguity and variation found in the training data, but typically this method cannot be used on its own to suggest variants due to dialect or other phonological processes that are not present in the training data.

9.2. Sound-to-letter conversion

Only relatively few publications have addressed the inverse problem of grapheme-to-phoneme conversion, namely inferring the correct spelling of a word from its phonemic transcription. Examples include the works of Meng et al. (1994) and Galescu and Allen (2002). An advantage of the method proposed here is that it uses statistical models that are symmetric with respect to both sides of the transduction. It is thus straight forward to apply them in the opposite direction.

Table 10 states some results to illustrate the level of accuracy that can be achieved. On the English NETtalk data set we find the spelling accuracy to be relatively close to the pronunciation accuracy on the reverse task. The letter error rate is 10% higher that the phoneme error rate. At 76%, this ratio is significantly worse on CMUdict. This discrepancy can be explained by the higher number of proper names, abbreviations and acronyms in CMUdict. A high accuracy on German shows again that this language has a rather phonetic orthography. In contrast to this, the phoneme-to-grapheme accuracy on the French data set is strikingly poor. This can be attributed to the notoriously high number of homophones in this language and in particular to the prevalence of silent final consonants.

Table 10
Phoneme-to-grapheme conversion accuracy for selected data sets

Language	Data set		Letter error rate [%]	Word error rate [%]
US English	NETtalk 18k	Galescu	10.03	41.87
		This work	8.62 ± 0.16	37.27 ± 0.57
US English	CMUdict	Galescu	11.5	49.7
		This work	10.35 ± 0.22	47.31 ± 0.73
German	LexDb	This work	0.41 ± 0.04	2.86 ± 0.22
French	Brulex	This work	5.62 ± 0.11	26.75 ± 0.44

Comparable results from Galescu and Allen (2002) are quoted.

Combining sound-to-letter conversion with a phoneme recognizer to orthographically transcribe out-of-dictionary words suggests itself. However, unguided acoustic phoneme recognition is notoriously inaccurate (Bisani and Ney, 2001), which impedes the applicability of phoneme-to-grapheme conversion in speech transcription.

9.3. Facilitate creation of pronunciation dictionaries

The techniques described in this article were used during the creation of the German LC-STAR lexicon for speech synthesis and recognition (Ziegenhain, 2005; Bisani et al., 2005). This lexicon contains information about over 46,000 proper names and 54,000 common words. For each entry part-of-speech information and phonemic transcriptions in SAMPA notation (Wells, 1997a; Wells, 1997b) are provided including syllabification and stress marks.

Creating the phonemic transcriptions in this lexicon was facilitated by using the data-driven statistical grapheme-to-phoneme converter described here. The phonemic transcriptions have to indicate both syllable boundaries and stress. Therefore, we simply added a syllable boundary marker as well as a stress marker to the phoneme inventory. Our method has no provisions to account for the structural properties of stress and syllabification. Therefore, structural errors occur, e.g. multiple primary stresses in a word or syllables without a nucleus. Nevertheless, we found that our G2P method provided reasonably good syllable boundary and stress prediction. Structural errors are not harmful in this application, because they can be filtered out very easily.

Initially, the most frequent words were transcribed manually. Then a grapheme-to-phoneme conversion model was estimated and additional words were transcribed automatically. These automatic transcripts were manually verified and corrected and then added to the G2P training data yielding an improved model. These steps were iterated several times. Over several iterations the fraction of incorrect pronunciations found in the verification step was reduced below 1%. In early iterations we selected which words to

verify randomly. In later iterations we computed the orthographic perplexity of each word, which is defined as $p(\boldsymbol{g})^{-1/|\boldsymbol{g}|}$, using an M-gram model $p(\boldsymbol{g})$ based on the G2P training data available at that point. The words with the highest perplexity were chosen for manual correction. These are hardest for the G2P model to predict correctly, because they correspond least to the words seen in training. However, the reader should be warned that there is a danger in overdoing this. Odd words may accumulate in the training data in higher than natural concentration, causing the algorithm to prefer the exception over the rule.

9.4. Spotting errors in pronunciation databases

When manually revising a pronunciation lexicon it is very helpful to have some guidance of where to look for wrong entries. We have examined the following criteria for finding incorrect entries in the G2P output:

- *Orthographic probability*: $p(\boldsymbol{g})$
  This measure takes into account that the G2P model typically gives bad results for words that are very unlike the words seen in the training set. Dissimilarity with the training data is reflected by a low probability.
- *Orthographic perplexity*: $p(\boldsymbol{g})^{-1/|\boldsymbol{g}|}$
- *Phonotactic probability*: $p(\boldsymbol{\varphi})$
  Phonotactic probability measures how much the word sounds like a typical word of the language. Strange unpronounceable phoneme sequences should have a low phonotactic probability.
- *Phonotactic perplexity*: $p(\boldsymbol{\varphi})^{-1/|\boldsymbol{\varphi}|}$
- *Grapheme-to-phoneme posterior probability*: $p(\boldsymbol{\varphi}|\boldsymbol{g})$
  This criterion is theoretically most appealing as it corresponds to our estimate of the probability that $\boldsymbol{\varphi}$ is the correct pronunciation of $\boldsymbol{g}$.

A quantitative evaluation of these criteria has been carried out in the course of the LC-STAR German lexicon project. The grapheme-to-phoneme conversion model used in this experiment was trained on 31,405 manually verified example pronunciations, 2% of which were used as a held-out set, without fold-back. The model uses only singular graphones with an M-gram order of $M = 7$. Another 86 thousand words have been transcribed with this model and the afore mentioned criteria have been computed for each of them. From this list we have randomly selected 5746 words for manual verification. We did this by sampling uniformly from the range of observed values of the quantities studied, and picking the word with the closest value. This procedure allows us to judge error rate as a function of the confidence measure considered. Uniform sampling on a per entry basis, would have led to very poor coverage of the regions of very high or very low confidence. The list of entries was sorted alphabetically before it was given to a human expert for correction, so that the expert had no indication of whether a particular pronunciation had high or low confidence.

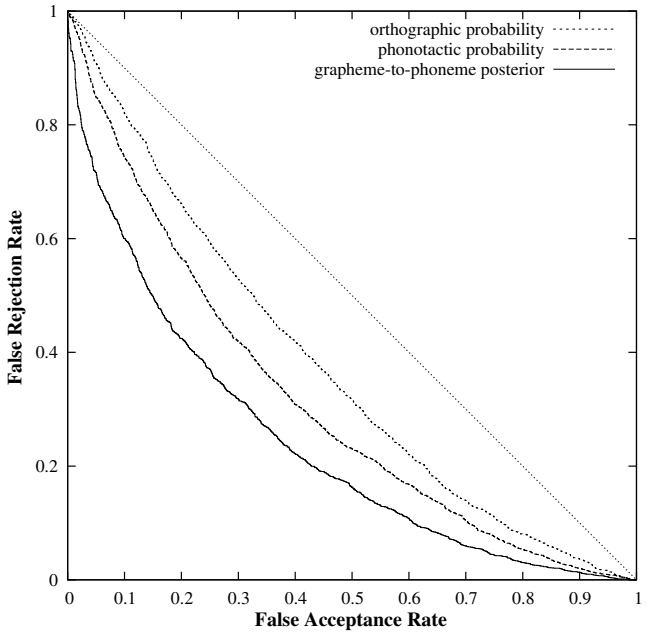


Fig. 6. Detection-error trade-off curves to grapheme-to-phoneme confidence measures.

Table 11
Comparison of different confidence measures for grapheme-to-phoneme conversion

Measure	Equal error rate [%]
Orthographic perplexity	41.7
Orthographic probability	41.1
Phonotactic perplexity	41.3
Phonotactic probability	35.6
Grapheme-to-phoneme posterior probability	31.2

Fig. 6 shows detection error trade-off curves for the three best measures considered. Equal error rates for all studied measures are given in Table 11. Equal error rate (also called cross-over error rate) is the operating point at which false acceptance and false rejection rates are equal. It is obvious that posterior probability performs much better than all of the other measures. Orthographic and phonotactic perplexity are consistently inferior to the corresponding probability. A possible explanation for this is that perplexity favors longer words because the impact of unlikely letters or phonemes is diluted. In contrast to this, our data indicate that longer words seem to have a higher chance of containing an error. Presumably this is simply because long words have more phonemes that can be wrong.

It should be emphasized that the pronunciations assessed were generated by the very same method and data that were used to verify them. This means that classification into correct and wrong entries is harder than finding gross mistakes or inconsistencies between different sources.

Acknowledgements

The work reported in this article was partially funded by the European Commission under Human Language Technologies Projects CORETEX (IST-1999-11876), LC-STAR (IST-2001-32216), and TC-STAR (IST-2002-FP6-506738).

References

Andersen, O., Kuhn, R., Lazaridès, A., Dalsgaard, P., Haas, J., Nöth, E., 1996. Comparison of two tree-structured approaches for grapheme-to-phoneme conversion. In: Proc. Internat. Conf. on Spoken Language Processing, Vol. 3, Philadelphia, PA, USA, pp. 1700–1703.

Bagshaw, P.C., 1998. Phonemic transcription by analogy in text-to-speech synthesis: novel word pronunciation and lexicon compression. Computer Speech Lang. 16, 119–142.

Bellegarda, J.R., 2005. Unsupervised, language-independent grapheme-to-phoneme conversion by latent analogy. Speech Comm. 46 (2), 140–152.

Besling, S., 1994. Heuristical and statistical methods for grahpeme-to-phoneme conversion. In: Konferenz zur Verarbeitung natürlicher Sprache (KONVENS), Vienna, Austria, pp. 24–31.

Bisani, M., Jolles, F., Popovic, M., 2005. LC-Star German lexicon for speech synthesis and recognition. Available from: European Language Resources Association, Catalog Reference S0245.

Bisani, M., Ney, H., 2001. Breadth-first search for finding the optimal phonetic transcription from multiple utterances. In: Proc. European Conf. on Speech Communication and Technology, Vol. 2, Aalborg, Denmark, pp. 1429–1432.

Bisani, M., Ney, H., 2002. Investigations on joint-multigram models for grapheme-to-phoneme conversion. In: Proc. Internat. Conf. on Spoken Language Processing, Vol. 1, Denver, CO, USA, pp. 105–108.

Bisani, M., Ney, H., 2003. Multigram-based grapheme-to-phoneme conversion for LVCSR. In: Proc. European Conf. on Speech Communication and Technology, Vol. 2, Geneva, Switzerland, pp. 933–936.

Bisani, M., Ney, H., 2004. Bootstrap estimates for confidence intervals in ASR performance evaluation. In: Proc. IEEE Internat. Conf. on Acoustics, Speech and Signal Processing, Vol. 1, Montreal, Canada, pp. 409–411.

Bisani, M., Ney, H., Sep. 2005. Open vocabulary speech recognition with flat hybrid models. In: Proc. European Conf. on Speech Communication and Technology, Lisbon, Portugal, pp. 725–728.

Caseiro, D., Trancoso, I., Oliveira, L., Viana, C., 2002. Grapheme-to-phone using finite-state transducers. In: Proc. IEEE Workshop on Speech Synthesis, Santa Monica, CA, USA.

Celex, 1995. The CELEX lexical database. <http://www.kun.nl/celex/>.

Chen, S.F., 2003. Conditional and joint models for grapheme-to-phoneme conversion. In: Proc. European Conf. on Speech Communication and Technology, Geneva, Switzerland, pp. 2033–2036.

Chen, S.F., Goodman, J., 1999. An empirical study of smoothing techniques for language modeling. Computer Speech Lang. 13 (4), 359–394, Oct.

Chen, S.F., Rosenfeld, R., 2000. A survey of smoothing techniques for ME models. IEEE Trans. Speech Audio Process. 8 (1), 37–50.

Content, A., Mousty, P., Radeau, M., 1990. Brulex: Une base de données lexicales informatisée pour le français écrit et parlé. In: L'Année Psychologique, pp. 551–566.

Daelemans, W., van den Bosch, A., 1996. Language-independent data-oriented grapheme-to-phoneme conversion. In: Van Santen, J.P.H., Sproat, R.W., Olive, J.P., Hirschberg, J. (Eds.), Progress in Speech Synthesis. Springer Verlag, Berlin, New York, pp. 77–90.

Dedina, M.J., Nusbaum, H.C., 1991. PRONOUNCE: a program for pronunciation by analogy. Computer Speech Lang. 5 (1), 55–63.

Deligne, S., Bimbot, F., 1995. Language modeling by variable length sequences: theoretical formulation and evaluation of multigrams. In: Proc. IEEE Internat. Conf. on Acoustics, Speech and Signal Processing, Vol. 1, Detroit, MI, USA, pp. 169–172.

Deligne, S., Bimbot, F., 1997. Inference of variable-length acoustic units for continuous speech recognition. Speech Comm. 23, 223–241.

Deligne, S., Yvon, F., Bimbot, F., 1995. Variable-length sequence matching for phonetic transcription using joint multigrams. In: Proc. European Conf. on Speech Communication and Technology, Madrid, Spain, pp. 2243–2246.

Galescu, L., 2003. Recognition of out-of-vocabulary words with sub-lexical language models. In: Proc. European Conf. on Speech Communication and Technology, Geneva, Switzerland, pp. 249–252.

Galescu, L., Allen, J.F., 2001. Bi-directional conversion between graphemes and phonemes using a joint *n*-gram model. In: Proc. 4th ISCA Tutorial and Research Workshop on Speech Synthesis, Perthshire, Scotland.

Galescu, L., Allen, J.F., 2002. Pronunciation of proper names with a joint *n*-gram model for bi-directional grapheme-to-phoneme conversion. In: Proc. Internat. Conf. on Spoken Language Processing, Vol. 1, Denver, CO, USA, pp. 109–112.

Gollan, C., Bisani, M., Kanthak, S., Schlüter, R., Ney, H., 2005. Cross domain automatic transcription on the TC-Star EPPS corpus. In: Proc. IEEE Internat. Conf. on Acoustics, Speech and Signal Processing, Vol. 1, Philadelphia, PA, USA, pp. 825–828.

Häkkinen, J., Suontausta, J., Riis, S., Jensen, K.J., 2003. Assessing text-to-phoneme mapping strategies in speaker independent isolated word recognition. Speech Comm. 41 (2), 455–467.

Jensen, K.J., Riis, S., 2000. Self-organizing letter code-book for text-to-phoneme neural network model. In: Proc. Internat. Conf. on Spoken Language Processing, Vol. 3, Beijing, China, pp. 318–321.

Jiang, L., Hon, H.-W., Huang, X., 1997. Improvements on a trainable letter-to-sound converter. In: Proc. European Conf. on Speech Communication and Technology, Vol. 2, Rhodes, Greece, pp. 605–608.

Kaplan, R.M., Kay, M., 1994. Regular models of phonological rule systems. Comput. Linguist. 20 (3), 331–378.

Kingsbury, P., Strassel, S., McLemore, C., MacIntyre, R., 1997. CALLHOME American English lexicon (PRONLEX). LDC Catalog No. LDC97L20.

Kneser, R., Ney, H., 1995. Improved backing-off for *M*-gram language modeling. In: Proc. IEEE Internat. Conf. on Acoustics, Speech and Signal Processing, Vol. 1, Detroit, MI, USA, pp. 181–184.

Levenshtein, V.I., 1965. Binary codes capable of correcting deletions, insertions and reversals. Dokl. Akad. NAUK SSSR 163 (4), 845–848, also in Cybernet. Control Theory, 10 (8), 707–710, 1966 (in Russian).

Lucassen, J.M., Mercer, R.L., 1984. An information theoretic approach to the automatic determination of phonetic baseforms. In: Proc. IEEE Internat Conf. on Acoustics, Speech and Signal Processing, Vol. 9, San Diego, CA, USA, pp. 304–307.

Lüngen, H., Ehlebracht, K., Gibbon, D., Simões, A.P.Q., 1998. Bielefelder Lexikon und Morphologie in VERBMOBIL Phase II. Technical Report ISSN 1434-8845, Universität Bielefeld.

Lööf, J., Bisani, M., Gollan, C., Heigold, G., Hoffmeister, B., Plahl, C., Schlnter, R., Ney, H., 2006. The 2006 RWTH parliamentary speeches transcription system. In: Proc. Internat. Conf. on Spoken Language Processing, Pittsburgh, PA, USA, pp. 105–108.

Marchand, Y., Damper, R.I., 2000. Multistrategy approach to improving pronunciation by analogy. Comput. Linguist. 26 (2), 195–219.

McCulloch, N., Bedworth, M., Bridle, J., 1987. NETspeak – a re-implementation of NETtalk. Computer Speech Lang. 2 (3–4), 289–302.

Meng, H.M., Seneff, S., Zue, V.W., 1994. Phonological parsing for bi-directional letter-to-sound/sound-to-letter generation. In: HLT'94: Proc. Workshop on Human Language Technology. Association for Computational Linguistics, Morristown, NJ, USA, pp. 289–294.

Mitton, R., 1992. Computer-usable dictionary file based on the Oxford Advanced Learner's Dictionary of Current English. <http://ota.ahds.ac.uk/>.

Ney, H., Essen, U., Kneser, R., 1995. On the estimation of small probabilities by leaving-one-out. IEEE Trans. Pattern Anal. Machine Intell. 17 (12), 1202–1212.

Ney, H., Martin, S., Wessel, F., 1997. Statistical language modeling using leaving-one-out. In: Young, S., Bloothoft, G. (Eds.), Corpus-based Methods in Language and Speech Processing. Kluwer Academic Publishers, Dordrecht, The Netherlands, pp. 174–207.

Och, F.J., Ney, H., 2003. A systematic comparison of various statistical alignment models. Comput. Linguist. 29 (1), 19–51.

Pagel, V., Lenzo, K., Black, A.W., 1998. Letter-to-sound rules for accented lexicon compression. In: Proc. Internat. Conf. on Spoken Language Processing, Vol. V, Sydney, Australia, pp. 2015–2018.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P., 1992. Numerical Recipes in C. Cambridge University Press (Chapter 10.5).

Robinson, T., 1997. BEEP – British English example pronunciations, version 1.0. <ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/dictionaries/beep.tar.gz>.

Sejnowski, T.J., Rosenberg, C.R., 1987. Parallel networks that learn to pronounce English text. Complex Systems 1 (1), 145–168.

Sejnowski, T.J., Rosenberg, C.R., 1993. NETtalk corpus. <ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/dictionaries>.

Suontausta, J., Häkkinen, J., 2000. Decision tree based text-to-phoneme mapping for speech recognition. In: Proc. Internat. Conf. on Spoken Language Processing, Beijing, China.

Torkkola, K., 1993. An efficient way to learn English grapheme-to-phoneme rules automatically. In: Proc. IEEE Internat. Conf. on Acoustics, Speech and Signal Processing, Vol. 2, Minneapolis, MN, USA, pp. 199–202.

van den Bosch, A., Chen, S.F., Daelemans, W., Damper, R.I., Gustafson, K., Marchand, Y., Yvon, F., 2006. Pascal letter-to-phoneme conversion challenge. <http://www.pascal-network.org/Challenges/PRONALSYL>.

Vozila, P., Adams, J., Lobacheva, Y., Thomas, R., 2003. Grapheme to phoneme conversion and dictionary verification using graphonemes. In: Proc. European Conf. on Speech Communication and Technology, Geneva, Switzerland, pp. 2469–2472.

Weide, R.L., 1998. The Carnegie Mellon pronouncing dictionary. <http://www.speech.cs.cmu.edu/cgi-bin/cmudict>.

Wells, J.C., 1997a. SAMPA computer readable phonetic alphabet. <http://www.phon.ucl.ac.uk/home/sampa>.

Wells, J.C., 1997b. SAMPA computer readable phonetic alphabet. In: Gibbon, D., Moore, R., Winski, R. (Eds.), Handbook of Standards and Resources for Spoken Language Systems. Mouton de Gruyter, Berlin and New York (Chapter IV.B).

Yvon, F., 1996. Grapheme-to-phoneme conversion using multiple unbounded overlapping chunks. In: Proc. Conf. on New Methods in Natural Language Processing, Ankara, Turkey, pp. 218–228.

Ziegenhain, U., 2005. Creation of lexica for speech recognition and synthesis. LC-Star project deliverable D3.1 + D3.2 available from www.lc-star.com.