8

A Survey of Discriminative Language Modeling Approaches for Large Vocabulary Continuous Speech Recognition

Brian Roark

Discriminative training of language models has been the focus of renewed research interest recently, and significant improvements to high accuracy large vocabulary continuous speech recognition have been demonstrated through the use of such models. In this chapter, we review past and present work on this topic, and focus in particular on three key issues: training data, learning algorithms and features. We will show how simple models trained under standard approaches can provide significant accuracy gains with negligible impact on decoding time, and how richer models can provide further accuracy gains. We will argue that while the published results are promising, many practical issues remain under-explored, so that further improvements might be expected as the best practices become more developed.

8.1 Introduction

It has long been the case that system improvements in statistical speech recognition have come far more from acoustic modeling innovations than from language modeling innovations – to the extent that some top-notch researchers have quit working on language modeling for speech recognition, sometimes with a parting message of warning to future researchers against expecting system improvements. (See the extended version of (Goodman 2001),

Automatic Speech and Speaker Recognition: Large Margin and Kernel Methods Joseph Keshet and Samy Bengio
© 2009 John Wiley & Sons, Ltd

which can be found with the search engine query: 'language modeling' 'all hope abandon.') There are a number of reasons why little progress was observed due to language modeling, including (though certainly not confined to): extreme sparse data; optimization of objectives only very loosely corresponding to the true system objective; and difficulty integrating disparate potential features into a single model. Sparse data is a problem that can be alleviated with increased training data, which some may not consider a particularly interesting solution, though it can require solving difficult problems that arise when scaling up algorithms to handle larger data sets. The continued improvement of performance in speech recognition and machine translation due to language model training on extremely large training sets (Emami *et al.* 2007; Popat *et al.* 2007) illustrates the extent to which sparse data has been impacting language models. Even what have been considered very large corpora in the past do not provide adequate training for multinomial n-gram models with millions or even billions of parameters. In Popat *et al.* (2007), for example, the models have hundreds of billions of parameters (corresponding to n-gram features of various orders), which are trained on corpora of several trillion tokens; they find a linear gain with each doubling of the training data, with no indication of a plateau.

In this chapter, we will focus on language modeling improvements in large vocabulary continuous speech recognition (LVCSR) that are not due to an increase in the amount of training data, but rather to improvements in the other two dimensions of the problem that were identified in the previous paragraph: optimization of parameters with respect to objectives more closely related to actual system objectives; and the inclusion of a large number of heterogeneous feature sets derived from the output string hypotheses.

Neither of these two considerations is new to research in language modeling for LVCSR. It was suggested in Jelinek (1996) to train an 'acoustic-sensitive' language model whose parameters would be trained to minimize the expected uncertainty of the spoken text given the acoustic signal. Discriminative training methods that were successfully pursued for acoustic modeling, such as MMIE (Bahl *et al.* 1986) or related error-corrective training methods (Bahl *et al.* 1993; Juang *et al.* 1997), spurred initial research into discriminative modeling approaches such as Chen *et al.* (2000); Stolcke and Weintraub (1998); Stolcke *et al.* (2000), with modest utility. In addition, generative maximum entropy models have been explored for combining diverse features within a single model (Khudanpur and Wu 2000; Rosenfeld *et al.* 2001), including n-grams, topic labels and syntactic features. Thus the notion of exploiting diverse feature sets is not new to the field; and neither are discriminative approaches to parameter estimation.

Over the past few years, however, there has been a renewed effort in this direction, driven in large part by developments in the machine learning community. The benefits of global conditional modeling for sequence learning (Lafferty *et al.* 2001) and the application of large margin methods to sequence processing through the Perceptron algorithm (Collins 2002; Freund and Schapire 1999) have provided a unifying framework within which many related methods can be pursued and compared. The learning and inference algorithms associated with some of these approaches are efficient enough to be scaled up, so that models making use of very large feature sets can be effectively trained and used for inference. Well-motivated and theoretically grounded regularization techniques have been shown to effectively control for the overtraining that language models have long suffered from. Strong empirical gains have been realized on difficult tasks and continue to be realized up to the present within this paradigm.

This chapter addresses discriminative language modeling approaches in general, not kernel approaches in particular. The approaches that are the focus of this chapter do not represent the only strand of research focused on leveraging competing LVCSR system hypotheses to achieve reductions in error rate. The approaches we describe here are focused on *training* models with a discriminative objective function, not *inference* techniques for optimizing an objective. Hence approaches such as Minimum Bayes-Risk classification (Goel and Byrne 2000) or confusion networks ('sausages') (Mangu *et al.* 2000) will not be covered here, although methods exist for learning their edit distance (Shafran and Byrne 2004)[1]. Note that inference using such techniques can also be pursued with models trained as presented here, hence the techniques are complementary. We will also omit any detailed discussion of methods that perform post-processing on LVCSR output to correct specific errors, such as induction of transformational rules on the output (Mangu and Padmanabhan 2001) or methods for proposing alternatives that are not present in the system output and choosing between them (Ringger and Allen 1996; Zhou *et al.* 2006b). Again, such methods could also be applied to the output of systems using models trained with methods presented in this survey, hence they are complementary to what will be discussed here. Also, we have chosen to focus on large vocabulary systems, hence will not discuss the application of similar methods in small vocabulary applications, e.g. that in Kuo *et al.* (2002).

8.2 General Framework

Most of the work covered in this chapter falls within the general framework of linear models, which we present here largely following the notation of Collins (2002), much as presented in (Roark *et al.* 2007). These approaches assume the presence of several components, as follows.

First, these are supervised approaches, hence requiring the presence of training data, consisting of a set of m input sequences $\bar{x}_i \in \mathcal{X}$ and corresponding reference (ground truth) output sequences $\bar{y}_i^{\text{ref}} \in \mathcal{Y}$, where $\mathcal{X}$ denotes the set of possible inputs and $\mathcal{Y}$ the set of possible outputs. In the case of language modeling for LVCSR, using a vocabulary Σ, the input set $\mathcal{X}$ consists of acoustic samples, the output set consists of all possible transcriptions, i.e. $\mathcal{Y} = \Sigma^*$, and the reference (ground truth) output sequence for any given utterance is the manual (human) transcription of that utterance.

Next, for each input sequence $\bar{x} \in \mathcal{X}$, the approaches require some mechanism for identifying a set of output candidates, which is often denoted as a function $\mathbf{GEN}(\bar{x}) \subseteq \mathcal{Y}$. For example, $\mathbf{GEN}(\bar{x})$ could be the word lattice output or n-best list from a baseline LVCSR system.

Third, there must be a mapping from each input:output pairing $(\bar{x}, \bar{y}) \in \mathcal{X} \times \mathcal{Y}$ to a real valued d-dimensional feature vector $\boldsymbol{\psi}(\bar{x}, \bar{y}) \in \mathbb{R}^d$, for some d. In the current case, this could be a mapping from a string to the set of n-grams of order $\leq k$ that occur in the string. An example feature would then be something like 'feature 1236: number of times the bigram "dog house" occurs in the string'. Each explicit n-gram to be included in the model would be a separate feature in the vector.

[1]The methods for learning the edit distance used in such techniques are focused on learning a weighted transduction between reference transcripts and LVCSR system outputs. This learned transduction can be seen as a form of discriminative language modeling, since it effectively moves the centroid closer to the reference.

Finally, there must be a real valued parameter vector $\boldsymbol{\alpha} \in \mathbb{R}^d$ of the same dimension d as the feature vector. For example, feature 1236 mentioned above may have a parameter value of 0.2 assigned to it. Methods for determining the parameter values associated with each feature are perhaps the key difference between many of the different approaches that will be presented here, though they are all discriminative rather than generative approaches.

Given these four components, we can define the best scoring (or 1-best) output $\bar{y}_i^{\max}$ for a given input x_i as

$$\bar{y}_i^{\max} = \arg \max_{\bar{y} \in \mathbf{GEN}(\bar{x}_i)} \boldsymbol{\psi}(\bar{x}_i, \bar{y}) \cdot \boldsymbol{\alpha}, \tag{8.1}$$

where $\boldsymbol{\psi}(\bar{x}_i, \bar{y}) \cdot \boldsymbol{\alpha}$ is the dot product.

In the remainder of this section, we will focus on each of these four components in turn, highlighting some of the key trends and variations in the work from this area.

8.2.1 Training Data and the GEN Function

Training data for discriminative language modeling consists of input speech recordings and reference output strings representing the gold standard transcription of the speech. The role of the GEN function is typically taken by a baseline LVCSR system, so that GEN($\bar{x}$) for some input speech is an n-best list or word lattice. Hence most discriminative language modeling approaches fall within the range of approaches more traditionally known as n-best or lattice rescoring. Working from lists and lattices, however, is not a requisite part of the approach, and some recent papers (Kuo *et al.* 2007; Lin and Yvon 2005) have modified the weights directly within the baseline recognition system, rather than adopting a rescoring or reranking paradigm. See section 8.2.2 for further details on directly updating the baseline recognition system, where it is discussed as a method for encoding features to allow for efficient processing.

The output of LVCSR systems can be thought of as weighted, acyclic finite state automata, with words and weights labeling the arcs. Figure 8.1 shows three kinds of representation of weighted output from an LVCSR system. For all such weighted automata, the semiring must be specified, so that the weights can be interpreted. The automata in Figure 8.1 are in the Real semiring, meaning that the weights along a path are multiplied, and when two paths are combined, their weights are added, which is the appropriate semiring when dealing with probabilities[2]. In the figure, the weights define a conditional probability distribution over three possible strings: 'a b c' with probability 0.3; 'a g c', with probability 0.3; and 'e d c c', with probability 0.4. Although the weights are presented as conditional probabilities, they can be arbitrary.

Not all weighted finite-state automata (WFA) can be determinized, but any acyclic WFA can be determinized (Mohri and Riley 1997). The result of determinization on the n-best list automaton in Figure 8.1(a) is the deterministic tree in Figure 8.1(b). As with the original representation, a candidate string is represented by the concatenation of symbols labeling the arcs of any path from the start state at the left to a final state denoted with a double

[2]The most common semirings in speech processing are the Tropical and Log semirings, both of which are appropriate for negative log probabilities. Both add weights along the path. The Tropical semiring takes the **min** of two weights when combining paths, and the Log semiring takes the log of the sum of the exponentials: $\mathrm{A} \oplus \mathrm{B} = -\log(\exp(-\mathrm{A}) + \exp(-\mathrm{B}))$.

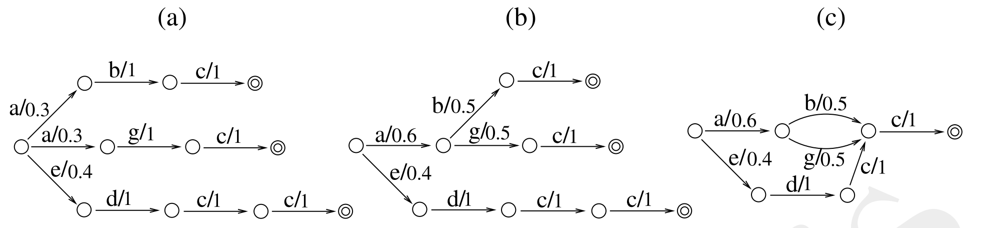

Figure 8.1 Three weighted, acyclic finite-state automata representations of the output from a speech recognizer: (a) n-best list; (b) deterministic tree; (c) deterministic word lattice. The automata are in the Real semiring, and all three represent the same weighted distributions over the three strings.

circle. What is key to notice from these figures is that determinization preserves the path weight of every path in the automaton. Recall that weights are combined along a path in the Real semiring by multiplying. Inspection will verify that all three of the paths in the original automaton are present in the deterministic tree representation with the same path weight.

Deterministic WFA can be minimized, i.e. converted to a representation of exactly the same weighted paths, but with a minimal number of states and arcs to represent it (Mohri and Riley 1997). For the current example, this minimized WFA is shown in Figure 8.1(c). These are word lattices, which are simply a more compact representation of a set of distinct weighted hypotheses output from the LVCSR system. Efficient use of this kind of compact representation allows for effective rescoring in scenarios where brute force enumeration of all distinct candidates would be intractable. Effective use of these representations, however, precludes the use of global features, unless they can be decomposed for dynamic programming. Some approaches to discriminative language modeling, e.g. those making use of long-distance dependencies uncovered via context-free parsing, will be forced to operate on undeterminized or unminimized n-best representations, which limits the number of candidates that can be tractably scored.

There are a couple of reasons to introduce WFA representations in this context. First, they become important in the next section, when efficient feature mapping is discussed. Also, it should be clear from the discussion that there is no difference between n-best lists and word lattices, other than determinization and minimization – they both represent sets of competing (weighted) candidate transcriptions. Even 1-best output of a system is a special case of these acyclic WFA. Hence we can discuss the approaches in general, without having to specify whether the data is in the form of n-best lists or lattices. We will thus henceforth refer to them simply as lattices, which may or may not be determinized or minimized. General operations on WFA can be used to specify operations on the lattices, including intersection, composition with a transducer, and finding the minimum cost (or highest scoring) path.

Note that working within a reranking or rescoring paradigm has a few immediate ramifications for the general modeling framework we are presenting. Most critically, it is not guaranteed in this approach that the reference transcription is present in the lattice. Second, if training lattices are produced for utterances that were also used for building the models of the baseline recognizer, this may introduce a mismatch between training conditions and testing conditions. These are two key issues investigated in Roark *et al.* (2004, 2007), and we discuss them here in turn.

(a) Reference Transcript

If the reference transcript for a particular utterance is not in the lattice output in the training data, then standard parameter estimation algorithms will be trying to achieve an objective that cannot be reached. These parameter estimation algorithms will be discussed in section 8.2.3, but, intuitively, an algorithm (such as the Perceptron algorithm) that penalizes candidates other than the gold standard (reference) will move parameters to penalize every candidate if the gold standard is not in the set of candidates, even the highest accuracy candidate in the whole set.

One quick fix to this problem would be to append the reference transcript to the set of candidates in the training set. The main problem with this is that it creates a great mismatch between the training condition and the testing condition, in that the training examples are being given better candidate sets than will be seen under testing conditions. Of lesser significance is the problem of acquiring features derived from the baseline LVCSR system for the reference transcript, if it was not one of its candidates. For example, the baseline system score is typically a useful feature in the model. Obtaining the baseline system score for the reference may require having some access to the baseline system models, to constrain the decoder to output the reference.

A better solution is to define the gold standard candidate in terms of maximum accuracy or minimum error rate. Then, if the reference is in the candidate set, it will be the gold standard; otherwise, another candidate from the set is chosen as the gold standard, to avoid the problems with parameter estimation. When the reference is present in the lattice, the gold standard is unique. If the reference is not present in the lattice, then there may not be a unique gold standard: more than one candidate may have the minimum error rate. The results in Roark *et al.* (2004, 2007) empirically validate the utility of using a minimum error rate gold standard versus a reference gold standard in the Perceptron algorithm. In those papers, one gold standard is chosen from among the set of minimum error rate candidates; in (Singh-Miller and Collins 2007), all members from the set were collectively used as the gold standard in a loss-sensitive version of the Perceptron algorithm. See section 8.2.3 for more details about parameter estimation when the reference is not in the lattice.

Note that this is also an issue when training outside of a rescoring or reranking paradigm, as noted in Kuo *et al.* (2007), even with known training data. If the vocabulary is pruned, there will be out-of-vocabulary (OOV) terms in the references which the closed vocabulary LVCSR system cannot recognize. One method for decreasing the OOV rate in any of these training scenarios, however, is to map terms to ‘equivalent’ items that are in the vocabulary, e.g. legal spelling variations.

(b) Seen Versus Unseen Data

The second issue that was mentioned regarding the training data is the potential mismatch between training and testing scenarios when training lattices are produced on previously seen data. This is an issue with any kind of a reranking problem, and similar solutions to the one that will be discussed here have been used for other such problems, e.g. (Collins 2000). In fact, the problem is to minimize mismatch, because most problems have a constrained amount of supervised training data (with both input and output sides of the mapping), hence the same data will often be used both for training the baseline models and for training discriminative language models. It is well-known that n-gram language models are overparameterized and overtrained; the result of this is that the LVCSR output for seen data will have fewer errors than the output for unseen data, and those errors that do occur may differ from the kinds of error observed for unseen data. The discriminative reranking model will then learn to solve a different problem than the one that will be encountered at test time. To obtain a useful model, the mismatch needs to be reduced by producing training lattices in an unseen condition.

An unseen condition can be achieved through cross-validation, whereby the training data is split into k subsets, and the baseline models for each subset j are trained on the other $k-1$ subsets $(1, \ldots, j-1, j+1, \ldots, k)$. Note that this can still result in a mismatch, because the testing condition will use baseline models trained on all k subsets. The hope is that, if k is large enough, the difference in performance when training on the extra section will be relatively small, particularly compared with the difference in seen versus unseen performance. Again, the results in Roark *et al.* (2004, 2007) empirically validate the utility of cross-validating the baseline language model training: not cross-validating resulted in models that achieved no gain over the baseline system, whereas cross-validation produced significant word-error rate (WER) reductions of over 1% absolute.

While the presentation here has been on the importance of cross-validating the models used in the baseline systems, previous work has only examined the utility of cross-validating the language models, not the acoustic models. The reasons for this are twofold. First, conventional wisdom is that acoustic models, by virtue of having far fewer parameters than language models, tend not to overtrain to the same degree, so that the seen/unseen difference in performance is relatively small. Second, while baseline n-gram models are relatively easy and fast to train (even for large corpora), since they typically use relative frequency estimation and simple smoothing to estimate the parameters, acoustic model training is a far more intensive process, so that training a baseline acoustic model on a large training corpus may take many days or more on multiple processors. Hence the cost of cross-validation is high and the perceived benefit low. Even so, it would be of interest to know exactly how much mismatch exists between seen and unseen data with respect to the acoustic model, and whether this may actually impact the utility of training data for discriminative language modeling.

8.2.2 Feature Mapping

The third component of the general framework that we are presenting is the mapping from input/output pairs $(\bar{x}, \bar{y})$ to a feature vector $\boldsymbol{\psi}(\bar{x}, \bar{y}) \in \mathbb{R}^d$. In the current context, there are two scenarios that we have discussed: reranking from lattices; and updating the baseline LVCSR system directly. We will first discuss feature mapping for reranking, where candidates are represented as an acyclic WFA of candidates; then for direct update of the system.

For reranking, one approach to feature mapping is to simply perform feature mapping for each candidate transcript independently. For example, if an existing statistical parser were to be applied to the strings and the result used to extract features, then the particular parser may not be able to handle anything other than a string at a time. This requires explicit enumeration of all candidates, which will limit the number of candidates that can be considered for any given utterance.

A more efficient strategy is to take advantage of shared structure in a word lattice to extract features without having to explicitly enumerate every path in the lattice. This is particularly important in LVCSR, where the number of competing hypotheses can be in the hundreds of thousands or millions, and the feature set in the several millions. N-gram features are popular in large part because they have an efficient finite-state structure, and this structure can be exploited for efficient feature mapping that allows for large lattices to be rescored, rather than small n-best lists. We will use a toy example to illustrate this. For more details on n-gram model structure, see Allauzen *et al.* (2003) and Roark *et al.* (2004, 2007, 2004).

For simplicity, assume a vocabulary $\Sigma = \{a,b,c\}$, and all strings consist of one or more symbols from that vocabulary in sequence (Σ^+). The output symbols will be feature indices, which we will denote f_k for feature with index k. For this example, let us assume the following unigram and bigram features: $\{f_1 = a$; $f_2 = b$; $f_3 =$<s>a; $f_4 =$<s>b; $f_5 =$ aa; $f_6 =$ ab; $f_7 =$ a</s>; $f_8 =$ ba; $f_9 =$ ba; f_{10}=b</s>$\}$. Note that the begin-of-string symbol (<s>) and end-of-string symbol (</s>) are implicit in the string and not part of the vocabulary. One can see by inspection of this set that there are all unigram and bigram features including the symbols ‘a’ and ‘b’, but none including the symbol ‘c’, hence those n-grams will not map to any feature in the set.

The finite-state transducer in Figure 8.2(a) maps from strings in $\{a,b,c\}^+$ to feature indices from this set. The start state (denoted with $>$ to its left) implicitly encodes the begin-of-string symbol, hence the first symbol in the string will be in a bigram beginning with <s>. There are two special symbols in the transducer: ϕ, which represents a failure transition; and ϵ, which represents the empty string. An arc labeled with ϕ is traversed if and only if there is no arc leaving the state that is labeled with the next symbol in the input. No input symbols are consumed or output symbols emitted when the ϕ transition is traversed. An arc with an ϵ symbol labeling the input side can always be traversed without consuming an input symbol, while outputting the symbol on the output side. An arc with an ϵ symbol labeling the output side produces no output while consuming the symbol on the input side. When final states (denoted with double circle) are reached and there is no more input, the transduction is successful.

Let us step through a couple of examples. First the simple string ‘acb’, which we will compose with transducer T_1 from Figure 8.2(a). Beginning at the start state, there is an arc labeled with ‘a’ on the input side, with f_3 on the output side. From the destination state of that arc, there is no arc with ‘c’ on the input side, so the failure transition (labeled with ϕ) is traversed. From the destination state of that arc, there is an arc with ‘c’ on the input side, which is a looping arc with ϵ (empty string, i.e. no feature) on the output side. The transition from that state with ‘b’ on the input side has f_2 on the output side. The destination state of that arc is not a final state, but there is a transition with an ϵ on the input side which can be traversed, which gives f_{10} on the output side and reaches a final state. Hence this transduction gives the features: f_3, f_2 and f_{10}. This is not, however, the total feature set, but rather the highest order n-grams reached at each word. For example, feature f_3 is the bigram ‘<s>a’, but there should also be the unigram feature f_1 associated with the ‘a’. In order to derive the total feature set based on the highest order features, we compose the output sequence to the transducer T_2 in Figure 8.2(b). In that transducer, all features map to themselves, but additionally all bigram features ending in an ‘a’ or ‘b’ then also map to the appropriate unigram feature. The complete transduction from a string S to the feature set is $S \circ T_1 \circ T_2$, where $\circ$ denotes the composition of finite-state transducers.

Note that these are both deterministic on the input side, i.e. every string in $\{a,b,c\}^+$ has one and only one path through $T_1 \circ T_2$. It should be straightforward for the reader

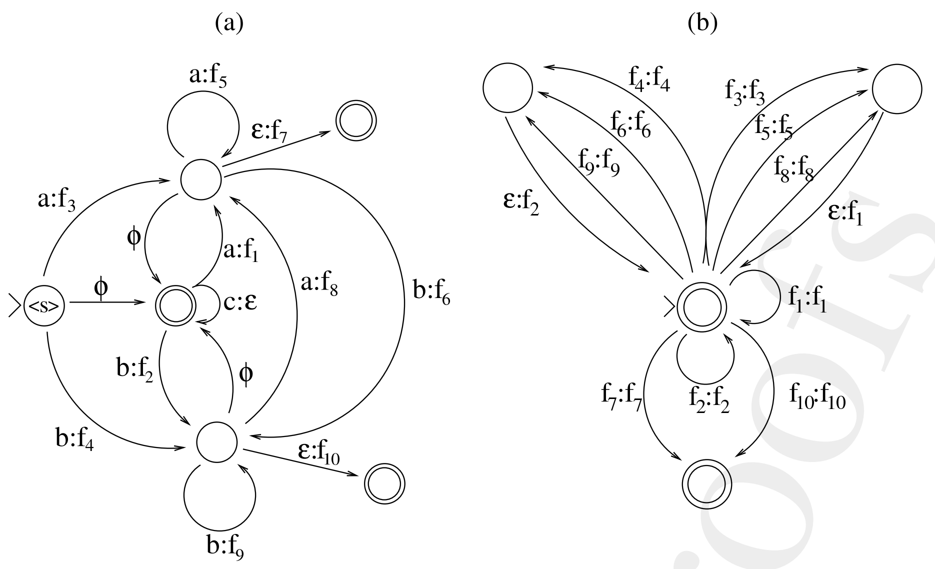

Figure 8.2 String sequence to feature set transducers: (a) from symbols to highest order features; (b) from highest order features to total set of features.

to verify that the input string 'bbacb', when composed with $T_1 \circ T_2$, yields the feature sequence $f_4f_2f_9f_2f_8f_1f_2f_{10}$. Because this feature mapping uses standard finite-state transducer transduction, word lattices encoded as finite-state automata can be composed with these transducers, to produce output sequences for every path in the lattice corresponding to features in the feature set. Multiple paths that share part of their structure will not need to be mapped independently of each other, as they would be in non-deterministic representations of *n*-best lists.

Note that, at test time, the feature identities are not of interest, but rather the scores of the paths in the lattices, so that the best scoring candidate can be found. There is also a compact finite-state automaton that can score the lattice with *n*-gram parameter weights, without requiring transduction to the feature labels. The automaton in Figure 8.3 is in the Tropical semiring (sum weights along a path), and makes use of roughly the same topology as the transducer in Figure 8.2(a). Note that all arcs corresponding to bigram features must have the sum of both the bigram feature weight and the corresponding unigram feature weight. Parameter weights associated with bigrams containing the end-of-string symbol are encoded in the final weight of the corresponding final state. Similar structure is used when going to *n*-grams of higher order than bigrams. See Roark *et al.* (2004, 2007, 2004) for more information about such finite-state *n*-gram model encoding.

Finite-state transducer encoding of the model is also a central topic in the recent work looking at direct updating of parameters in the baseline LVCSR model rather than reranking (Kuo *et al.* 2007; Lin and Yvon 2005). These works adopt an approach to efficient LVCSR that involves building a finite-state 'decoding graph' off-line and performing optimizations on this transducer, as described in detail in Mohri *et al.* (2002). The decoding graph is the

Idx	Param	Feature
1	$\boldsymbol{\alpha}_1$	a
2	$\boldsymbol{\alpha}_2$	b
3	$\boldsymbol{\alpha}_3$	<s>a
4	$\boldsymbol{\alpha}_4$	<s>b
5	$\boldsymbol{\alpha}_5$	aa
6	$\boldsymbol{\alpha}_6$	ab
7	$\boldsymbol{\alpha}_7$	a</s>
8	$\boldsymbol{\alpha}_8$	ba
9	$\boldsymbol{\alpha}_9$	bb
10	$\boldsymbol{\alpha}_{10}$	b</s>

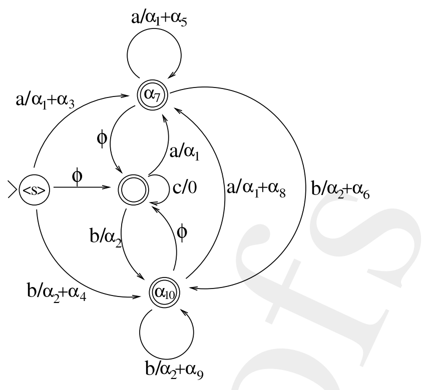

Figure 8.3 Deterministic automaton reading in symbols and assigning parameter weights.

result of combining an automata representation of an *n*-gram language model (G) with a pronunciation dictionary represented as a finite-state transducer (L) and decision tree models that split the phones in the pronunciations into context-dependent (CD) phone representations (C). The composition of these weighted transducers[3] $C \circ L \circ G$ results in a single transducer, which serves to guide decoding. Optimizations to this graph, such as determinization on the input side and weight pushing, can result in a far more efficient search than combining these models on-the-fly during search.

The idea pursued in Lin and Yvon (2005) and extended to full context-dependent LVCSR systems in Kuo *et al.* (2007) is to directly update transition weights within the decoding graph. To do this, they determine the state sequence through the decoding graph of the reference transcription by optimally force aligning the transcription with the speech. Training in Kuo *et al.* (2007) is an on-line procedure discussed in section 8.2.3 that compares the best scoring state sequence with the state sequence derived from the reference, and updates the transition weights accordingly. Hence each transition weight in the graph is an independent feature within a log linear framework.

This approach to feature mapping brings up some interesting questions. First, what is discriminative language modeling? In this instance, the features include context-dependent phone identities as part of the state definition, and stochastic modeling of such features is typically outside the range of language modeling. While it is not the intent of this chapter to try to philosophically delimit the scope of this topic, it seems clear that the transition weights in these graphs represent discrete symbol sequences rather than the Gaussian densities modeled at the states, hence are certainly modeling the language not the acoustics. However, it highlights the fact that there is little in this general framework to limit the kinds of features that might be used, and that future work may find hybrid feature sets – e.g. for combining syntactic and prosodic features – that will lead to substantial system improvements.

The second question that arises is, what do these features represent? The input stream of the transducer is labeled with context-dependent phones, and the output stream is labeled with words in the vocabulary, so in its simplest form, these features represent (CD phone $\times$

[3]Note that an automaton is a special case of a transducer where every symbol maps to itself.

word) sub-sequences. These graphs, however, are optimized to increase the amount of shared structure between words with similar CD phone realizations, hence the transitions themselves may represent CD phones over sets of *n*-grams. These complicated composite features do appear to have strong utility for this task (see Kuo *et al.* 2007 for empirical results), and they capture key distinctions that are clearly lost in most language modeling approaches. The downside of this approach is a loss of generalization from one phone sequence realization of a word sequence to another realization of the same word sequence. One could imagine an extension of this approach that treats arcs as features, but also includes raw *n*-gram features, and updates arcs in the graph with respect to all relevant features. In such a way, the benefits of this approach could be achieved while also gaining generalization to other pronunciations of the same *n*-gram.

8.2.3 Parameter Estimation

The final component to the general framework is the parameter estimation algorithm, for choosing a parameter weight vector $\boldsymbol{\alpha} \in \mathbb{R}^d$. Here any technique that has been used for log linear modeling in other domains can be used, with the proviso that the method must scale to very large problems. Language modeling for LVCSR is typically most effective with a large number of candidates (at least in the hundreds) and a very large number of features (in the millions). This means that there have been limited uses of resource intensive methods such as Support Vector Machines (SVMs) (though see section 8.3.3 for one example using SVMs).

Before getting into specific training objectives, it is probably worth spending a little time to address a potential confusion of terminology. In language modeling, *conditional* models have typically signified the conditional probabilities used via the chain rule to define the joint probability of a string. For example, a standard trigram model is defined as the probability of a word conditioned on the previous two words. In the current context, however, conditional models do not denote the local conditional probabilities that are used to define the joint generative probability, but rather denote the likelihood of the string *given* the input utterance. This sort of conditional model is discriminative.

Given the popularity of Maximum Mutual Information Estimation (MMIE) for acoustic modeling (Bahl *et al.* 1986), which has a conditional likelihood objective in the sense we are using here, direct applications of such techniques have been pursued for language modeling. As in prior work of Stolcke and Weintraub (1998), Chueh *et al.* (2005) examines a log linear model which maximizes an unregularized conditional likelihood objective. Linguistic (*n*-gram) features of the model are combined with a baseline acoustic model score, which can be thought of as just another feature in the model. If $\bar{y}_i^{\text{ref}}$ is the reference output for speech input $\bar{x}_i$, then the conditional log likelihood of the training data of m utterances given a feature mapping $\boldsymbol{\psi}$ and parameter vector $\boldsymbol{\alpha}$ is

$$\begin{aligned}\text{LogLike}(\boldsymbol{\alpha}) &= \sum_{i=1}^{m} \log \text{P}(\bar{y}_i^{\text{ref}} \mid \bar{x}_i) \\ &= \sum_{i=1}^{m} \log\left(\frac{\exp(\boldsymbol{\psi}(\bar{x}_i, \bar{y}_i^{\text{ref}}) \cdot \boldsymbol{\alpha})}{\sum_{\bar{y}\in\text{GEN}(\bar{x}_i)} \exp(\boldsymbol{\psi}(\bar{x}_i, \bar{y}) \cdot \boldsymbol{\alpha})}\right) \\ &= \sum_{i=1}^{m} \left(\boldsymbol{\psi}(\bar{x}_i, \bar{y}_i^{\text{ref}}) \cdot \boldsymbol{\alpha} - \log \sum_{\bar{y}\in\text{GEN}(\bar{x}_i)} \exp(\boldsymbol{\psi}(\bar{x}_i, \bar{y}) \cdot \boldsymbol{\alpha})\right). \qquad (8.2)\end{aligned}$$

This is a convex function, and hence the $\boldsymbol{\alpha}$ that maximizes the log likelihood can be found through standard techniques relying on the gradient (see below).

One key problem with this approach is overtraining, which, as usual with language modeling, is severe and has led to little or no system gain for approaches pursuing it as presented above (Chueh *et al.* 2005; Stolcke and Weintraub 1998). With millions of features, the opportunity for overtraining is much greater for language modeling than for acoustic modeling, hence some method for controlling for overtraining (regularization) needs to be added to the training approach in order to improve generalization to new data. Of course, the reasons why these models provided little or no gain in system performance may also be related to some of the issues in training data preparation presented in section 8.2.1, but certainly lack of regularization also contributed. Similar lack of system improvement was observed in Paciorek and Rosenfeld (2000) when using Minimum Classification Error (MCE) training (Juang *et al.* 1997) for language model training, as well as in (Kuo and Chen 2005) when using Minimum Word Error (MWE) training (Povey and Woodland 2002). Direct application of popular acoustic model training approaches have thus not been successful, and a large reason for this is almost certainly lack of appropriate model regularization.

Adopting regularization techniques from other sequence modeling work making use of global conditional likelihood optimization (Johnson *et al.* 1999; Lafferty *et al.* 2001), the regularized objective in Roark *et al.* (2007, 2004) becomes:

$$\mathrm{RegLogLike}(\boldsymbol{\alpha}) = \sum_{i=1}^{m}\bigg(\boldsymbol{\psi}(\bar{x}_i, \bar{y}_i^{\mathrm{ref}}) \cdot \boldsymbol{\alpha} - \log \sum_{\bar{y}\in \mathrm{GEN}(\bar{x}_i)} \exp(\boldsymbol{\psi}(\bar{x}_i, \bar{y}) \cdot \boldsymbol{\alpha}) - \frac{\| \boldsymbol{\alpha} \|^2}{2\sigma^2}\bigg), \tag{8.3}$$

which is the common zero-mean Gaussian regularizer that imposes a penalty within the objective as the magnitude of the parameter values grow. This is also a convex function, thus guaranteeing convergence to the globally optimal solution. The derivative of this function with respect to any given parameter $\boldsymbol{\alpha}_k$ is

$$\frac{\partial\ \mathrm{RegLogLike}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}_k} = \sum_{i=1}^{m}\bigg(\boldsymbol{\psi}_k(\bar{x}_i, \bar{y}_i^{\mathrm{ref}}) - \sum_{\bar{y}\in \mathrm{GEN}(\bar{x}_i)} \mathrm{P}(\bar{y} \mid \bar{x}_i)\boldsymbol{\psi}_k(\bar{x}_i, \bar{y})\bigg) - \frac{\boldsymbol{\alpha}_k}{\sigma^2}. \tag{8.4}$$

The empirical results in Roark *et al.* (2007, 2004) validate that the unregularized global conditional likelihood objective in Equation (8.2) did not lead to any system improvements, while the regularized objective in Equation (8.3) does provide statistically significant improvements over the baseline, using an empirically chosen regularization parameter σ. The gradients of the *n*-gram features in that work were efficiently extracted from large word lattices using general finite-state techniques, which enabled the approach to be scaled up to over 10 million features, using lattices encoding a very large number of candidate transcriptions, for a training corpus consisting of hundreds of hours of speech.

Another technique adopted from the recent machine learning literature is the averaged Perceptron (Collins 2002; Freund and Schapire 1999), which was shown to provide significant word-error rate reductions over a baseline LVCSR system in Roark *et al.* (2004) and was later compared with global conditional likelihood optimization in Roark *et al.* (2007, 2004). The reader is referred to the cited papers for a formal presentation of the algorithm. Informally, this is an on-line algorithm that initializes parameters at zero and updates the parameters after every example as follows: using the current model, find the best scoring candidate of the current example ($\bar{y}_i^{\max}$); then update the parameters by penalizing features extracted from $\bar{y}_i^{\max}$ and rewarding features extracted from $\bar{y}_i^{\text{ref}}$, the gold standard. Thus, the parameter update at example $(\bar{x}_i, \bar{y}_i^{\text{ref}})$, in its simplest form, is

$$\boldsymbol{\alpha}[i] = \boldsymbol{\alpha}[i-1] + \boldsymbol{\psi}(\bar{x}_i, \bar{y}_i^{\text{ref}}) - \boldsymbol{\psi}\left(\bar{x}_i, \arg\max_{\bar{y}\in\mathbf{GEN}(\bar{x}_i)} \boldsymbol{\psi}(\bar{x}_i, \bar{y}) \cdot \boldsymbol{\alpha}[i-1]\right), \tag{8.5}$$

where $\boldsymbol{\alpha}[i-1]$ is the parameter vector from the previous time step[4]. Of course, if $\bar{y}_i^{\text{ref}} = \arg\max_{\bar{y}\in\mathbf{GEN}(\bar{x}_i)} \boldsymbol{\psi}(\bar{x}_i, \bar{y}) \cdot \boldsymbol{\alpha}[i-1]$, then the parameters are unchanged.

Like the conditional likelihood optimization, this approach is prone to overtraining, hence requires some regularization. The most common technique used for these models is *averaging* the parameters over the models at every timestep of the algorithm. The averaged parameter $\text{avg}_i(\boldsymbol{\alpha}_k)$ at time i is defined as

$$\text{avg}_i(\boldsymbol{\alpha}_k) = \frac{1}{i}\sum_{j=1}^{i} \boldsymbol{\alpha}_k[j]. \tag{8.6}$$

The raw parameter values are used while training the model, but this averaged parameter value is then used for evaluating the model on held aside or test data. Typically, some held-aside data is evaluated after every iteration over the training data, to determine when to stop training.

One benefit of Perceptron training versus regularized conditional likelihood optimization is that the Perceptron training typically arrives at a more parsimonious solution, i.e. the number of features with non-zero parameter weights are fewer, at least with commonly used regularizers. The reason for this is clear – parameters can only be updated in the Perceptron algorithm if the feature is extracted from either the gold standard candidate $\bar{y}_i^{\text{ref}}$ or the best scoring candidate $\bar{y}_i^{\max}$, hence many features never move away from their initial values of zero. In the comparisons between Perceptron and conditional likelihood optimization in Roark *et al.* (2007, 2004), the conditional likelihood estimation resulted in larger gains over the baseline than the Perceptron estimation, but the Perceptron was useful as an initial step prior to the conditional likelihood optimization, both for initializing parameter weights for fast convergence and for selecting features to include in the model.

Recent work has looked at improving on Perceptron language model training by introducing loss-sensitivity to the on-line parameter update in Equation (8.5). In the approach of Singh-Miller and Collins (2007), the contribution of an example to the change in the parameter values is based on a loss value, which in the cited paper is taken as the number of errors beyond the minimum error rate hypotheses in the n-best lists. Features extracted from highly errorful hypotheses will be more heavily penalized. In addition, this approach provides a well-motivated way of dealing with multiple candidates with the same minimum error rate in the set of candidates. The previously discussed (section 8.2.2) work in Kuo *et al.* (2007) also uses an on-line parameter estimation approach related to the Perceptron algorithm, where the magnitude of the parameter value update is determined via a sigmoid function.

While the above-mentioned parameter estimation techniques make use of well behaved (convex) functions, the objectives are not precisely what is typically evaluated, which is

[4]The algorithm may perform multiple passes over the training data, so the index i may indicate the number of updates, rather than the absolute example number in the corpus.

word-error rate (WER): the number of insertions, deletions or substitutions per 100 words of reference text. The global conditional likelihood approach is maximizing the (regularized) conditional likelihood of the gold standard; and the Perceptron is optimizing the string error rate rather than the WER. MWE training (Povey and Woodland 2002) optimizes a continuous function more closely related to error rate than either of these two objective functions, so perhaps a regularized version of that would be worth investigating. Others have tried methods of optimizing the WER function directly, which cannot be done with any guarantees of convergence or finding a global optimum. For example, minimum sample risk training (Gao *et al.* 2005) performs a greedy search of parameterizations by performing a line search along each dimension in sequence. Although no guarantees are given for finding an optimal solution, reported empirical results (Gao *et al.* 2005) are competitive with other methods such as Boosting and the averaged Perceptron. Similarly unguaranteed parameter search techniques were used in Banerjee *et al.* (2003) to update bigram parameter weights for word-error reduction.

We have presented a basic, general characterization of how these sorts of problems can be approached, and have pointed to specific papers pursuing specific versions of this general approach. Some of these have achieved relatively large improvements (more than 1% absolute) over baseline LVCSR systems, using just n-gram features. In the following section, we will survey some extensions to the basic approach, in three key directions: novel features; novel objective functions; and within the context of domain adaptation.

8.3 Further Developments

8.3.1 Novel Features

Up to now, we have discussed models that rely on n-grams as features alongside the baseline model score, or treat arcs within a decoding graph as features, e.g. Kuo *et al.* (2007). In this section, we point to three relatively recent papers that look to expand upon these simple feature sets to achieve additional discrimination. Note that this sort of enterprise has been notoriously difficult in generative language modeling for LVCSR, where WER reductions due to features other than n-grams have been hard to come by.

Syntactic language models have been a topic of some interest over the past decade, with some notable successes in achieving WER reductions, e.g. Charniak (2001); Chelba and Jelinek (2000); Roark (2001); Wang *et al.* (2003). In these works, generative context-free (Charniak 2001; Chelba and Jelinek 2000; Roark 2001) or finite-state (Wang *et al.* 2003) models are used to define a distribution over strings that can be used as a language model. Within a discriminative language modeling framework, however, Collins *et al.* (2005) used a generative context-free parsing model not to derive a score, but rather to derive a syntactic annotation which was then used within the feature mapping $\boldsymbol{\psi}$ for discriminative modeling with the Perceptron algorithm. Example features included part-of-speech (POS) and shallow parse sub-sequence features; context-free rule instances in the parses; and bilexical head-to-head dependencies. This stands in marked contrast to the generative models, where the generative model scores were used for modeling and the structural annotations were an unused byproduct. The WER reductions in Collins *et al.* (2005) were statistically significant (though modest) versus just using n-gram features, and most of the reduction was achieved with POS-tag sequence features.

A similar approach was taken for incorporating morphological features into language models in Czech (Shafran and Hall 2006). Unlike generative morphology-based language models, which have been used profitably in, for example, Arabic speech recognition (Vergyri *et al.* 2004), the use of the kinds of sub-word features that morphological analysis affords does not necessitate complicated smoothing techniques. Rather, the annotations from morphological analyzers can be used within the feature mapping to provide additional features in the model. This is particularly important in highly inflected or agglutinative languages, and promises substantial improvements in such languages, as demonstrated in Shafran and Hall (2006).

One key issue in discriminative syntactic and morphological language modeling approaches is feature selection. Features can be derived from rich structural annotations in many ways, which can lead to an astronomical number of potential features in the model. Features were selected in Shafran and Hall (2006) via a χ^2 statistic derived from the co-occurrence of features and the classes of interest, in this case 'correct' and 'incorrect' candidates. A large χ^2 statistic indicates a high correlation between a feature and either the set of correct candidates or the set of incorrect candidates, hence indicating that the feature would have high discriminative utility.

Trigger features (Lau *et al.* 1993) are a way of capturing the bursty and topical nature of language – if a word is observed, then there is typically a high likelihood of that word occurring again or related words occurring. A self-trigger is a feature that exploits an instance of a word or term to increase the likelihood of observing another, or in the case of discriminative models, to reward candidates that have the triggered term. In Singh-Miller and Collins (2007), self-triggers of various sorts were used within a loss-sensitive Perceptron modeling framework to reduce WER significantly over using just *n*-gram features. To do so, they define a history h of previous utterances in the conversation, and derive trigger features based on unigrams and bigrams that either a) occur multiple times in the candidate transcription; or b) occur in the candidate transcription and in transcriptions in the history h. In addition, they create backoff bins of unigrams and bigrams based on TF-IDF statistics, and derive another set of features based on these bins, as a way of tying the parameters for words that are similarly distributed. The combination of raw and binned trigger features, along with standard *n*-gram features, resulted in the best performing system.

Real gains are being achieved by expanding the feature set beyond simple *n*-gram features, and the discriminative language modeling paradigm that we have been presenting is a natural framework for work on novel methods for feature extraction and selection.

8.3.2 Novel Objectives

While most research in LVCSR over the years has focused on WER as an objective, it is increasingly becoming the case that speech recognition is a sub-process within a larger application such as call-routing, spoken language understanding or spoken document retrieval. Within such a scenario, discriminative language modeling approaches can be applied with a downstream objective function in training rather than an objective function based on the accuracy of the speech recognition itself. This sort of optimization can occur within the context of generative language model estimation, as with the spoken language understanding applications in Goel (2004); Goel *et al.* (2005), in which bigram probabilities were iteratively re-estimated to improve utterance annotation (Goel 2004) or call-routing

(Goel *et al.* 2005) performance. Discriminative language modeling methods, however, allow for great flexibility of feature mapping along with direct optimization of such an objective.

In Saraclar and Roark (2005, 2006), utterances within a call-routing application were annotated with class labels (97 classes), and discriminative language model estimation was done using either the Perceptron algorithm (Saraclar and Roark 2005) or regularized global conditional likelihood maximization (Saraclar and Roark 2006) for various objective functions: a WER objective; an utterance class error rate (CER) objective; and a joint WER/CER objective. Features included *n*-grams, class labels, and combination class label and *n*-gram. Optimization of the joint objective led to a very slight (not statistically significant) increase of CER versus training with an exclusively CER objective, but no increase in WER was observed versus training with an exclusively WER objective. That is, the trade-off inherent in a joint optimization came at the expense of CER rather than WER, which is unsurprising given that each utterance consists of many words but receives just one class label. Inclusion of the utterance class in the model achieved modest (though statistically significant) improvements in WER versus just *n*-gram features, hence this can also be seen as a method for increasing the feature set.

Given the growing interest in machine translation (MT) of speech, one might expect future work in pushing MT objective functions back into LVCSR model training, and discriminative language modeling of the sort described here is one way to achieve this. For example, the presence or absence of phrases from a phrase table within a phrase-based MT system may impact the final quality of translation, hence optimizing the LVCSR system to do a better job of recognizing such phrases, perhaps at the expense of overall WER, which may result in an overall system improvement. Similar approaches could be pushed in speech IR or other areas **Q64** of spoken language understanding. Other than the papers discussed here, we are unaware of any published results in this direction.

8.3.3 Domain Adaptation

All of the work discussed up to now has been within the context of in-domain trained systems, but language model adaptation to novel domains is an area of great interest where they have also been shown to have utility. An adaptation scenario does change a couple of key aspects of the training that impact the best practices. First, to the extent that the new domain is not part of the baseline training data (which is the whole point of domain adaptation) then the issue of cross validating the training set to get training data is moot. However, there remains the issue of the reference transcription not being in the lattice output of the recognizer. In fact, the oracle accuracy of the lattice, i.e. the best possible accuracy achievable from among the candidates, will likely be low for the novel domain, which can reduce the efficacy of these discriminative approaches, relative to generative alternatives such as Maximum a Posteriori (MAP) adaptation (Bacchiani *et al.* 2006; Gauvain and Lee 1994).

In Bacchiani *et al.* (2004), a large vocabulary general voicemail transcription system was adapted for customer service support voicemail recognition. The baseline out-of-domain training of roughly 100 hours was augmented with 17 hours of in-domain training data. In a straight head-to-head comparison, MAP adaptation of the generative language model, based on counts derived from the in-domain data, led to significantly larger improvements in transcription accuracy over the baseline (nearly 8% absolute) than using the Perceptron algorithm (5% absolute). Note that the vocabulary was kept constant in this application, so the

MAP adaptation did not benefit from additional in-domain vocabulary. The hypothesis was that the re-ranking nature of the Perceptron was hurting it, since the MAP adapted generative model was used in the first pass, hence could find good candidate transcriptions that were not in the word lattices produced by the baseline out-of-domain trained recognizer.

To remedy this, a hybrid of MAP adaptation followed by Perceptron reranking was pursued in Bacchiani *et al.* (2004). Of course, once the in-domain data was included in the baseline LVCSR training data, appropriate cross-validation techniques had to be used to produce training lattices for the discriminative language model. The result of this approach was an additional 0.7% absolute improvement versus using MAP adaptation alone.

In Zhou *et al.* (2006), several discriminative training algorithms were compared for domain adaptation in Chinese LVCSR, including the Perceptron algorithm, boosting (Schapire *et al.* 1998), ranking SVMs (Joachims 2002) and minimum sample risk (Gao *et al.* 2005, see section 8.2.3). They found the Perceptron algorithm provided the best domain adaptation (by more than 1% absolute), which they attributed to the large size of the feature set chosen by the Perceptron algorithm relative to the other methods investigated. In addition, they found that the ranking SVM took two orders of magnitude more training time than the other methods.

What remain unexplored are semi-supervised adaptation techniques for this problem – making use of in-domain text without recordings or recordings without transcriptions to adapt language models using discriminative parameter estimation techniques. The flexibility of generative adaptation approaches, such as MAP estimation, permits the opportunistic exploitation of available resources for model training. Achieving such flexibility within discriminative language modeling approaches is an important topic that deserves further attention.

8.4 Summary and Discussion

In this chapter, we have presented a general discriminative modeling framework and its application to language modeling for LVCSR. Nearly all of the work on discriminative language modeling fits this general characterization, and it is useful for defining the dimensions within which approaches differ. We have made an effort to be comprehensive in the survey, within the scope declared at the outset of the chapter. Overall the number of papers that have been written on this topic are relatively few, which can be attributed to a number of factors.

First, there are significant barriers to entry in researching this topic. Unlike other application areas, such as machine translation, achieving state-of-the-art performance in LVCSR is not accessible to researchers outside of a small number of research labs and centers. The number of techniques that must be combined within a full-blown LVCSR system to achieve this performance – including a range of acoustic modeling techniques in multiple stages, first-pass large vocabulary decoding, speaker adaptation and lattice rescoring techniques – are such that a researcher must expect a major time investment to even produce the training data needed to explore discriminative language modeling. What should be a priority within the speech community is to lower these barriers by making a large amount of training and evaluation word lattices available for research purposes to the community at large. That would almost certainly spur researchers in machine learning and NLP to begin applying their work in this domain.

It used to be the case that perplexity reduction was the primary mode for evaluating language model improvements. Given the poor correlation between reductions in perplexity and WER reduction, this is no longer widely accepted as a language modeling objective. It did have the virtue, however, of allowing evaluation on text corpora, which meant that research on language modeling could be conducted without access to LVCSR systems. For the discriminative models presented here, this is no longer the case: LVCSR systems are required for training data, and WER is the objective – perplexity cannot be calculated from these models. One step towards increasing research on language modeling would be to make word lattices from competitive LVCSR systems available to the community at large.

Even if such data were made available, however, this remains a very large and difficult problem. Scaling up machine learning algorithms to handle problem spaces with many millions of dimensions and training sets with millions or billions of examples (or more) is a challenge. There is a reason for discussing finite-state methods in this survey, and that is the scale of the problem. Coupled with the widespread perception that language modeling for LVCSR is hopeless, these challenges present another kind of barrier to entry.

This is a shame, because there has been progress in language modeling in this direction, and there remain a large number of very interesting and relatively unexplored directions in which to take it, such as the topics in the following short list:

- Semi-supervised techniques for leveraging large text corpora alongside the kind of training data we have been discussing;
- Further research on richer feature sets, such as syntactic features or hybrid acoustic/language features;
- Feature selection and/or dimensionality reduction;
- Scaling up compute intensive parameter estimation techniques.

This survey suggests that it is not unreasonable to foster the hope that real LVCSR system improvements can and will be achieved along these lines over the next decade, to the point where resulting techniques will become part of the collective best practices in the field.

References

Q65

Allauzen C, Mohri M and Roark B 2003 Generalized algorithms for constructing language models. *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 40–47.

Bacchiani M, Riley M, Roark B and Sproat R 2006 MAP adaptation of stochastic grammars. *Computer Speech and Language* **20**(1), 41–68.

Bacchiani M, Roark B and Saraclar M 2004 Language model adaptation with MAP estimation and the perceptron algorithm. *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pp. 21–24.

Bahl L, Brown P, de Souza P and Mercer R 1986 Maximum mutual information estimation of hidden Markov model parameters for speech recognition. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 49–52.

Bahl L, Brown P, de Souza P and Mercer R 1993 Estimating hidden Markov model parameters so as to maximize speech recognition accuracy. *IEEE Transactions on Speech and Audio Processing* **1**(1), 77–83.

Banerjee S, Mostow J, Beck J and Tam W 2003 Improving language models by learning from speech recognition errors in a reading tutor that listens. *Proceedings of the 2nd International Conference on Applied Artificial Intelligence*, Fort Panhala, Kolhapur, India.

Charniak E 2001 Immediate-head parsing for language models. *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Chelba C and Jelinek F 2000 Structured language modeling. *Computer Speech and Language* **14**(4), 283–332.

Chen Z, Lee KF and Li MJ 2000 Discriminative training on language model. *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*.

Chueh C, Chien T and Chien J 2005 Discriminative maximum entropy language model for speech recognition. *Proceedings of the European Conference on Speech Communication and Technology (Interspeech)*.

Collins M 2002 Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1–8.

Collins M, Saraclar M and Roark B 2005 Discriminative syntactic language modeling for speech recognition. *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 507–514.

Collins MJ 2000 Discriminative reranking for natural language parsing. *Proceedings of the 17th International Conference on Machine Learning*.

Emami A, Papineni K and Sorensen J 2007 Large-scale distributed language modeling. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Freund Y and Schapire R 1999 Large margin classification using the perceptron algorithm. *Machine Learning* **3**(37), 277–296.

Gao J, Yu H, Yuan W and Xu P 2005 Minimum sample risk methods for language modeling. *Proceedings of the Conference on Human Language Technology Conference and Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pp. 209–216.

Gauvain JL and Lee CH 1994 Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains. *IEEE Transactions on Speech and Audio Processing* **2**(2), 291–298.

Goel V 2004 Conditional maximum likelihood estimation for improving annotation performance of n-gram models incorporating stochastic finite state grammars. *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*.

Goel V and Byrne W 2000 Minimum Bayes-risk automatic speech recognition. *Computer Speech and Language* **14**(2), 115–135.

Goel V, Kuo H, Deligne S and Wu C 2005 Language model estimation for optimizing end-to-end performance of a natural language call routing system. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. I/565–568.

Goodman J 2001 A bit of progress in language modeling. *Computer Speech and Language* **15**(4), 403–434.

Jelinek F 1996 Acoustic sensitive language modeling. Center for Language and Speech Processing, Johns Hopkins University, Baltimore, MD. Technical report.

Joachims T 2002 Optimizing search engines using clickthrough data. *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

Johnson M, Geman S, Canon S, Chi Z and Riezler S 1999 Estimators for stochastic “unification-based” grammars. *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 535–541.

Juang B, Chou W and Lee C 1997 Minimum classification error rate methods for speech recognition. *IEEE Transactions on Speech and Audio Processing* **5**(3), 257–265.

Khudanpur S and Wu J 2000 Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling. *Computer Speech and Language* **14**(4), 355–372.

Kuo H, Kingsbury B and Zweig G 2007 Discriminative training of decoding graphs for large vocabulary continuous speech recognition. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. IV/45–48.

Kuo HKJ, Fosler-Lussier E, Jiang H and Lee CH 2002 Discriminative training of language models for speech recognition. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Orlando, FL.

Kuo J and Chen B 2005 Minimum word error based discriminative training of language models. *Proceedings of the European Conference on Speech Communication and Technology (Interspeech).*

Lafferty J, McCallum A and Pereira F 2001 Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of the 18th International Conference on Machine Learning*, pp. 282–289.

Lau R, Rosenfeld R and Roukos S 1993 Trigger-based language models: a maximum entropy approach. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 45–48.

Lin S and Yvon F 2005 Discriminative training of finite-state decoding graphs. *Proceedings of the European Conference on Speech Communication and Technology (Interspeech).*

Mangu L and Padmanabhan M 2001 Error corrective mechanisms for speech recognition. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP).*

Mangu L, Brill E and Stolcke A 2000 Finding consensus in speech recognition: word error minimization and other application of confusion networks. *Computer Speech and Language* **14**(4), 373–400.

Mohri M and Riley M 1997 Weighted determinization and minimization for large vocabulary speech recognition. *Proceedings of the European Conference on Speech Communication and Technology (Interspeech).*

Mohri M, Pereira FCN and Riley M 2002 Weighted finite-state transducers in speech recognition. *Computer Speech and Language* **16**(1), 69–88.

Paciorek C and Rosenfeld R 2000 Minimum classification error training in exponential language models. *Proceedings of the NIST Speech Transcription Workshop.*

Popat TBA, Xu P, Och F and Dean J 2007 Large language models in machine translation. *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 858–867.

Povey D and Woodland P 2002 Minimum phone error and I-smoothing for improved discriminative training. *Proceedings of the International Conference on Spoken Language Processing (ICSLP).*

Ringger EK and Allen JF 1996 Error corrections via a post-processor for continuous speech recognition. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP).*

Roark B 2001 Probabilistic top-down parsing and language modeling. *Computational Linguistics* **27**(2), 249–276.

Roark B, Saraclar M and Collins M 2004 Corrective language modeling for large vocabulary ASR with the perceptron algorithm. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. I/749–752.

Roark B, Saraclar M and Collins M 2007 Discriminative n-gram language modeling. *Computer Speech and Language* **21**(2), 373–392.

Roark B, Saraclar M, Collins M and Johnson M 2004 Discriminative language modeling with conditional random fields and the perceptron algorithm. *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 47–54.

Rosenfeld R, Chen S and Zhu X 2001 Whole-sentence exponential language models: a vehicle for linguistic-statistical integration. *Computer Speech and Language* **15**(1), 55–73.

Saraclar M and Roark B 2005 Joint discriminative language modeling and utterance classification. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. I/561–564.

Saraclar M and Roark B 2006 Utterance classification with discriminative language modeling. *Speech Communication* **48**(3-4), 276–287.

Schapire RE, Freund Y, Bartlett P and Lee WS 1998 Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics* **26**(5), 1651–1686.

Shafran I and Byrne W 2004 Task-specific minimum Bayes-risk decoding using learned edit distance. *Proceedings of the International Conference on Spoken Language Processing (ICSLP).*

Shafran I and Hall K 2006 Corrective models for speech recognition of inflected languages. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 390–398.

Singh-Miller N and Collins M 2007 Trigger-based language modeling using a loss-sensitive perceptron algorithm. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. IV/25–28.

Stolcke A and Weintraub M 1998 Discriminitive language modeling. *Proceedings of the 9th Hub-5 Conversational Speech Recognition Workshop*.

Stolcke A, Bratt H, Butzberger J, Franco H, Gadde VRR, Plauche M, Richey C, Shriberg E, Sonmez K, Weng F and Zheng J 2000 The SRI March 2000 Hub-5 conversational speech transcription system. *Proceedings of the NIST Speech Transcription Workshop.*

Vergyri D, Kirchoff K, Duh K and Stolcke A 2004 Morphology-based language modeling for Arabic speech recognition. *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*.

Wang W, Harper MP and Stolcke A 2003 The robustness of an almost-parsing language model given errorful training data. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Zhou Z, Gao J, Soong F and Meng H 2006 A comparative study of discriminative methods for reranking LVCSR n-best hypotheses in domain adaptation and generalization. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. I/141–144.

Zhou Z, Meng H and Lo W 2006 A multi-pass error detection and correction framework for Mandarin LVCSR. *Proceedings of the International Conference on Spoken Language Processing (ICSLP).*