Chapter 13
Automatic Speech Recognition

Hagen Soltau, George Saon, Lidia Mangu, Hong-Kwang Kuo, Brian Kingsbury, Stephen Chu, and Fadi Biadsy

13.1 Introduction

In this chapter we describe techniques to build a high performance speech recognizer for Arabic and related languages. The key insights are derived from our experience in the DARPA GALE program, a 5-year program devoted to enhancing the state-of-the-art in Arabic speech recognition and translation. The most important lesson is that general speech recognition techniques work very well also on Arabic. An example is the issue of vowelization: short vowels are often not transcribed in Arabic, Hebrew, and other Semitic languages. Semi-automatic vowelization procedures, specifically designed for the language, can improve the pronunciation lexicon. However, we also can simply choose to ignore the problem at the lexicon level, and compensate for the resulting pronunciation mismatch with the use of discriminative training of the acoustic models. While we focus on Arabic, in this chapter, we speculate that the vast majority of the issues we address here will completely carry over to other Semitic languages. We have tested the approaches discussed in this chapter only on Arabic, as that is the Semitic language with the most resources. Our experimental results demonstrate that such language-independent techniques can solve language-specific issues at least to a large extent. Another example is morphology, where we show that a combination of language-independent techniques (an efficient decoder to deal with large vocabulary and exponential language models) and language-specific techniques (a neural network language model that uses morphological and syntactic features) lead to good results.

H. Soltau (✉) • G. Saon • L. Mangu • H.-K. Kuo • B. Kingsbury • S. Chu
IBM T.J. Watson Research Center, Yorktown Heights, NY, USA
e-mail: hsoltau@us.ibm.com; aon@us.ibm.com; mangu@us.ibm.com; hkuo@us.ibm.com; bedk@us.ibm.com; schu@us.ibm.com

F. Biadsy
Google, New York, NY, USA
e-mail: fadi.biadsy@gmail.com

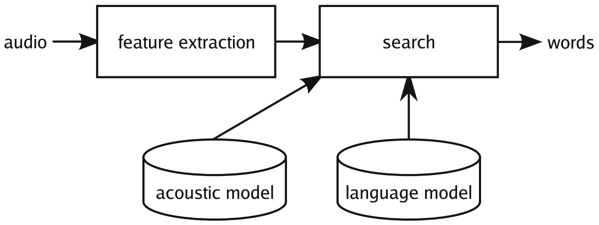

Fig. 13.1 Block diagram of an automatic speech recognition system

For these reasons we describe in the text a list of both language-independent and language-specific techniques. We describe also a full-fledged LVCSR system for Arabic that makes best use of all the techniques. We also demonstrate how this system can be used to bootstrap systems for related Arabic dialects and Semitic languages.

13.1.1 Automatic Speech Recognition

Modern speech recognition systems use a statistical pattern recognition approach to the problem of transforming speech signals to text. This approach is data-driven: to build a speech recognition system, practitioners collect speech and text data that are representative of a desired domain (e.g., news broadcasts or telephone conversations), use the collected data to build statistical models of speech signals and text strings in the target domain, and then employ a search procedure to find the best word string corresponding to a given speech signal, where the statistical models provide an objective function that is optimized by the search process. A high-level block diagram of such a speech recognition system is given in Fig. 13.1.

More precisely, in the statistical framework, the problem of speech recognition is cast as

$$\widehat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmax}} P(\mathbf{W}|\mathbf{X};\Theta) \tag{13.1}$$

where $\mathbf{W}$ is a word sequence, $\widehat{\mathbf{W}}$ is the optimal word sequence, $\mathbf{X}$ is a sequence of acoustic feature vectors, and Θ denotes model parameters.

Solving this problem directly is challenging, because it requires the integration of knowledge from multiple sources and at different time scales. Instead, the problem is broken down by applying Bayes' rule and ignoring terms that do not affect the optimization, as follows:

$$\widehat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmax}} P(\mathbf{W}|\mathbf{X};\Theta) \tag{13.2a}$$

$$= \underset{\mathbf{W}}{\operatorname{argmax}} \frac{P(\mathbf{X}|\mathbf{W};\Theta)P(\mathbf{W};\Theta)}{P(\mathbf{X};\Theta)} \tag{13.2b}$$

$$= \underset{\mathbf{W}}{\operatorname{argmax}} P(\mathbf{X}|\mathbf{W};\Theta)P(\mathbf{W};\Theta) \tag{13.2c}$$

Referring back to Fig. 13.1, we can identify different components of a speech recognition system with different elements of Eq. (13.2). The *feature extraction* module (Sect. 13.2.1) computes sequences of acoustic feature vectors, $\mathbf{X}$, from audio input. The *acoustic model* (Sect. 13.2.1) computes $P(\mathbf{X}|\mathbf{W};\Theta)$: the probability of the observed sequence of acoustic feature vectors, $\mathbf{X}$, given a hypothesized sequence of words, $\mathbf{W}$. The *language model* (Sect. 13.3.1) computes $P(\mathbf{W};\Theta)$, the prior probability of a hypothesized sequence of words. The *search* process (Sect. 13.3.2) corresponds to the argmax operator.

13.1.2 Introduction to Arabic: A Speech Recognition Perspective

An excellent introduction to the Arabic language in the context of ASR can be found in Kirchhoff et al. [27]. Here we describe only a couple of special characteristics of Arabic that may not be familiar to non-Arabic speakers: vowelization and morphology.

1. Vowelization
   Short vowels and other diacritics are typically not present in modern written Arabic text. Thus a written word can be ambiguous in its meaning and pronunciation. An Arabic speaker can resolve the ambiguity based on human knowledge and various contextual cues, such as syntactic and semantic information. Although Arabic automatic diacritizaion has received considerable attention by NLP researchers, the proposed approaches are still error-prone, especially on non-formal texts. When designing an ASR system, we therefore need to consider whether to represent words in the vowelized or un-vowelized form for the language model. Another consideration is whether the pronunciation model or dictionary should contain information derived from the diacritics, such as short vowels.
2. Morphology
   Arabic is a morphologically rich language. In Arabic morphology, most morphemes are comprised of a basic word form (the root or stem), to which affixes can be attached to form whole words. Arabic white-space delimited words may be then composed of zero or more prefixes, followed by a stem and zero or more

suffixes. Because of the rich morphology, the Arabic vocabulary for an ASR system can become very large, on the order of one million words, compared with English which typically has a vocabulary on the order of one hundred thousand words. The large vocabulary exacerbates the problem of data sparsity for language model estimation.

Arabic is the only Semitic language for which we have a sufficient amount of data, through the DARPA GALE program; therefore we have decided to focus on developing and testing our ASR techniques only for Arabic. We will describe how our design decisions helped us to successfully overcome some of the Arabic-dependent issues using new sophisticated language-dependent and independent modeling methods and good engineering.

It is important to note that the majority of the modeling techniques and algorithms outlined in this chapter have already been tested on a set of languages such as English, Mandarin, Turkish and Arabic. While we only discuss these ASR approaches for Arabic in this chapter, there is no reason to believe that these approaches would not work for other Semitic languages. The fundamental reason for this is that we were able to tackle language-specific problems with language-independent techniques. As an illustration, we also discuss in this chapter the similarities between the two Semitic languages Arabic and Hebrew. We speculate that the same language-dependent techniques used to address the challenges of Arabic ASR, such as the morphological richness of the language and diacritization would also work for Hebrew.

13.1.3 Overview

This chapter is organized as follows. In the first two sections, we describe the two major components for state-of-the-art LVCSR: the acoustic model and the language model. For each model, we distinguish between language-independent and language-specific techniques. The language-specific techniques for the acoustic models include vowelization and modeling of dialects in decision trees. The language-specific parts of the language model include a neural network model that incorporates morphological and syntactic features. In Sect. 13.4, we describe how all these techniques are used to build a full-fledged LVCSR system that achieves error rates below 10 % on an Arabic broadcast news task. In Sect. 13.5, we describe techniques that allow us to port Modern Standard Arabic (MSA) models to other dialects, in our case to Levantine. We describe a dialect recognizer and how this can be used to identify relevant training subsets for both the acoustic and language model. We describe a decoding technique that allows us to use a set of dialect-specific models simultaneously during run-time for improved recognition performance. Section 13.6 describes the various data sets we used for system training, development, and evaluation.

13.2 Acoustic Modeling

13.2.1 Language-Independent Techniques

Feature Extraction

The goal of the feature extraction module is to compute a representation of the audio input that preserves as much information about its linguistic content as possible, while suppressing variability due to other phenomena such as speaker characteristics or the acoustic environment. Moreover, this representation should be compact (typically generating about 40 parameters for every 10 ms of audio) and should have statistical properties that are compatible with the Gaussian mixture models most often used for acoustic modeling. A very common form of feature extraction is *Mel frequency cepstral coefficients* (MFCCs) [16], and most other approaches to feature extraction, such as perceptual linear prediction (PLP) coefficients, employ similar steps to MFCCs, so we describe their computation in detail below.

The steps for computing MFCCs are as follows.

1. The short-time fast Fourier transform (FFT) is used to compute an initial time-frequency representation of the signal. The signal is segmented into overlapping frames that are usually 20–25 ms in duration, with one frame produced every 10 ms, each frame is windowed, and a power spectrum is computed for the windowed signal.
2. The power spectral coefficients are binned together using a bank of triangular filters that have constant bandwidth and spacing on the Mel frequency scale, a perceptual frequency scale with higher resolution at low frequencies and lower resolution at high frequencies. This reduces the variability of the speech features without severely impacting the representation of phonetic information. The filter bank usually contains 18–64 filters, depending on the task, while the original power spectrum has 128–512 points, so significant data reduction takes place here.
3. The dynamic range of the features is reduced by taking the logarithm. This operation also means that the features can be made less dependent on the frequency response of the channel and on some speaker characteristics by removing the mean of the features over a sliding window, on an utterance-by-utterance basis, or for all utterances attributed to a given speaker.
4. The features are then decorrelated and smoothed by taking a low-order discrete cosine transform (DCT). Depending on the task, 13–24 DCT coefficients are retained.

In order to remove the effect of channel distortions, the cepstral coefficients are normalized so that they have zero mean and unit variance on a per utterance or a per speaker basis. The final feature stream includes the local temporal characteristics of the speech signal because these convey important phonetic information. Temporal context across frames can be incorporated by computing speed and acceleration

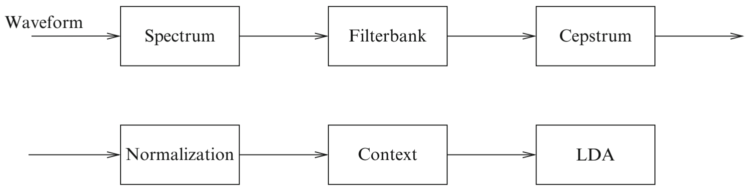

Fig. 13.2 Frontend pipeline

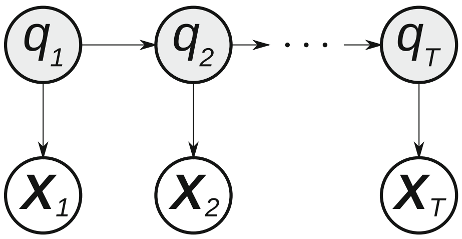

Fig. 13.3 Graphical model representation of a hidden Markov model

coefficients (or delta and delta-delta coefficients) from the neighboring frames within a window of typically ± 4 frames. These dynamic coefficients are appended to the static cepstra to form the final 39-dimensional feature vector. A more modern approach is to replace this ad-hoc heuristic with a linear projection matrix that maps a vector of consecutive cepstral frames to a lower-dimensional space. The projection is estimated to maximize the phonetic separability in the resulting subspace. The feature vectors thus obtained are typically modelled with diagonal covariance Gaussians. In order to make the diagonal covariance assumption more valid, the feature space is rotated by means of a global semi-tied covariance transform. This sequence of processing steps is illustrated in Fig. 13.2.

Acoustic Modeling

Speech recognition systems model speech acoustics using hidden Markov models (HMMs), which are generative models of the speech production process. A graphical model representation of an HMM is given in Fig. 13.3. An HMM assumes that a sequence of T acoustic observations $\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$ is generated by an underlying discrete-time stochastic process that is characterized by a single, discrete state q_t taking on values from an alphabet of N possible states. The state of the generative process is *hidden*, indicated by the shading in Fig. 13.3. The HMM also makes two important conditional independence assumptions. The first is that the state of the generating process at time t, q_t, is conditionally independent of all states and observations, given the state at the previous time step, q_{t-1}. The second assumption is that the acoustic observation at time t, $\mathbf{x}_t$, is conditionally independent of all states and observations, given q_t. These conditional independence assumptions are denoted by the directed edges in Fig. 13.3.

An HMM is specified by four elements: an alphabet of N discrete states for the hidden, generative process; a prior distribution over q_0, the initial state of the generative process; $P(q_t|q_{t-1})$, an $N \times N$ matrix of state transition probabilities; and $\{P(\mathbf{x}_t|q_t)\}$, a family of state-conditional probability distributions over the acoustic features. In most large-vocabulary speech recognition systems, the distribution of initial states is uniform over legal initial states. Similarly, the state transition matrix only allows a limited number of possible state transitions, but the distribution over allowable transistions is taken as uniform. Remaining are the methods used to define the state alphabet, which will be described in detail later in this section, and $\{P(\mathbf{x}_t|q_t)\}$, the observation probability distributions.

The standard approach to modeling the acoustic observations is to use Gaussian mixture models (GMMs)

$$P(\mathbf{x}_t|q_j) = \sum_{m=1}^{M} w_{mj}(2\pi)^{-\frac{k}{2}}|\boldsymbol{\Sigma}_{mj}|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}_t-\boldsymbol{\mu}_{mj})^T \boldsymbol{\Sigma}_{mj}^{-1}(\mathbf{x}_t-\boldsymbol{\mu}_{mj})} \tag{13.3}$$

where state q_j has M k-dimensional Gaussian mixture components with means $\boldsymbol{\mu}_{mj}$ and covariance matrices $\boldsymbol{\Sigma}_{mj}$, as well as mixture weights w_{mj} such that $\sum_m w_{mj} = 1$. Note that in most cases the covariance matrices are constrained to be diagonal. GMMs are a useful model for state-conditional observation distributions because they can model in a generic manner variability in the speech signal due to various factors, because their parameters can be estimated efficiently using the EM algorithm, and because their mathematical structure allows for forms of speaker and environmental adaptation based on linear regression.

To understand how the HMM state alphabet is defined, it is necessary to understand how words are modeled in large-vocabulary speech recognition systems. Speech recognition systems work with a finite, but large-vocabulary (tens to hundreds of thousands) of words that may be recognized. Words are modeled as sequences of more basic units: either phonemes or graphemes. Phonemes are the basic sound units of a language, and are thus a very natural compositional unit for word modeling. However, using phonemes entails a significant amount of human effort in the design of the speech recognition dictionary: somebody has to produce one or more phonetic pronunciations for every word in the dictionary. This cost has driven the use of graphemic dictionaries, in which words are modeled directly as sequences of letters. While this approach works well for some languages, such as Finnish, that have essentially phonetic spellings of words, it works less well for other languages. For example, consider the pair of letters "GH" in English, which can sound like "f" as in the word "enough," like a hard "g" as in the word "ghost," or can be silent as in the word "right."

To characterize the temporal structure of the basic speech units (phonemes or graphemes), each unit is modeled with multiple states. A popular choice for modeling is illustrated in Fig. 13.4, where the HMM topology is strictly left-to-right

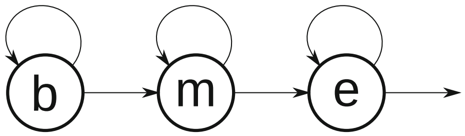

Fig. 13.4 A typical 3-state, *left*-to-*right* HMM, drawn as a finite-state machine. The beginning (*b*), middle (*m*), and end (*e*) of a basic speech unit are modeled separately to characterize the temporal structure of the unit

with self loops on the states. The 3-state model shown can have separate observation distributions for the beginning, middle, and end of the unit. If different units have different durations, these may be represented by allocating a greater or fewer number of states to different units.

A final, key ingredient in modeling of speech acoustics is the concept of context dependence. While phonemes are considered to be the basic units of speech, corresponding to specific articulatory gestures, the nature of the human speech apparatus is such that the acoustics of a phoneme are strongly and systematically influenced by the context in which they appear. Thus, the "AE" sound in the word "man" will often be somewhat nasalized because it is produced between two nasal consonants, while the "AE" in "rap" will not be nasalized. Context-dependence can also help with the ambiguity in going from spelling to sound in graphemic systems. Returning to the "GH" example from above, we know for English that a "GH" that occurs at the end of a word and is preceded by the letters "OU" is likely to be pronounced "f".

Although it produces more detailed acoustic models, context-dependence requires some form of parameter sharing to ensure that there is sufficient data to train all the models. Consider, for example, *triphone* models that represent each phone in the context of the preceding and following phone. In a naive implementation that represents each triphone individually, a phone alphabet of 40 phones would induce a triphone alphabet of $40^3 = 64{,}000$ triphones. If phones are modeled with 3-state, left-to-right HMMs as shown above, this would lead to $3 \times 64{,}000 = 192{,}000$ different models. Due to phonotactic constraints, some of these models would not occur, but even ignoring those, the model set would be too large to be practical.

The standard solution to this explosion in the number of models is to cluster the model set using decision trees. Given an alignment of some training data, defined as a labeling of each frame with an HMM state (e.g., the middle state of "AE"), all samples sharing the same label can be collected together and a decision tree can be grown that attempts to split the samples into clusters of similar samples at the leaves of the tree. The questions that are asked to perform the splits are questions about the context of the samples: the identities of the phonetic or graphemic units to the left and right; the membership of the neighboring units in classes such as "vowels," "nasals," or "stops;" and whether or not a word boundary occurs at some position to the left or right. A popular splitting criterion is data likelihood under a single, diagonal-covariance Gaussian distribution. In this case, the decision trees can be trained efficiently by accumulating single-Gaussian sufficient statistics for each context-dependent state, and then growing the decision trees. Once a forest of decision trees has been grown for all units, they are pruned to the desired size. Typically, a few thousand to ten thousand context-dependent states will be defined for a large-vocabulary speech recognition system.

The training process for speech recognition systems is usually iterative, beginning with simple models having few parameters, and moving to more complex models having a larger number of parameters. In the case of a new task, where there are no existing models that are adequately matched to it, speech recognition training begins with a *flat start* procedure. In a flat start, very simple models with no context-dependence and only a single Gaussian per state are initialized directly from the reference transcripts as follows. First, each transcript is converted from a string of words to a string of phones by looking up the words in the dictionary. If a word has multiple pronunciations, a pronunciation is selected at random. This produces a sequence of N phones. Next, the corresponding sequence of acoustic features is divided into N equal-length segments, and sufficient statistics for each model are accumulated from its segments. Once the models are initialized, they are refined by running the EM algorithm, and the number of Gaussian mixture components per state is gradually increased.

The models that are produced by a flat start are coarse, and usually have poor transcription accuracy; however, they are sufficient to perform *forced alignment* of the training data. In the forced alignment procedure, the reference word transcripts are expanded into a graph that allows for the insertion of silence between words and the use of the different pronunciation variants in the dictionary, and then the best path through the graph given an existing set of models and the acoustic features for the utterance is found using dynamic programming. The result of this procedure is an alignment of the training data in which every frame is labeled with an HMM state. Given an alignment, it is possible to train context-dependent models, as described above. Typically, for a new task, several context-dependent models of increasing size (in terms of the number of context-dependent HMM states and the total number of Gaussians) will be trained in succession, each relying on a forced alignment from the previous model.

For ASR systems we are interested in the optimality of the recognition accuracy however, and we aim to train the acoustic model discriminatively so as to achieve the lowest word error rate on unseen test data. Directly optimizing the word error rate is hard because it is not differentiable. Alternative approaches look at optimizing smooth objective functions related to word error rate (WER) such as minimum classification error (MCE), maximum mutual information (MMI) and minimum phone error (MPE) criteria. Discriminative training can be applied either to the model parameters (Gaussian means and variances) or to the feature vectors. The latter is done by computing a transformation called feature-space MPE (fMPE) that provides time-dependent offsets to the regular feature vectors. The offsets are obtained by a linear projection from a high-dimensional space of Gaussian posteriors which is trained such as to enhance the discrimination between correct and incorrect word sequences. Currently, the most effective objective function for model and feature-space discriminative training is called boosted MMI and is inspired by large-margin classification techniques.

Speaker Adaptation

Speaker adaptation aims to compensate for the acoustic mismatch between training and testing environments and plays an important role in modern ASR systems. System performance is improved by conducting speaker adaptation during training as well as at test time by using speaker-specific data. Speaker normalization techniques operating in the feature domain aim at producing a canonical feature space by eliminating as much of the inter-speaker variability as possible. Examples of such techniques are: vocal tract length normalization (VTLN), where the goal is to warp the frequency axis to match the vocal tract length of a reference speaker, and feature-space maximum likelihood linear regression (fMLLR), which consists in affinely transforming the features to maximize the likelihood under the current model. The model-based counterpart of fMLLR, called MLLR, computes a linear transform of the Gaussian means such as to maximize the likelihood of the adaptation data under the transformed model.

13.2.2 Vowelization

One challenge in Arabic speech recognition is that there is a systematic mismatch between written and spoken Arabic. With the exception of texts for beginning readers and important religious texts such as the Qur'an, written Arabic omits eight diacritics that denote short vowels and consonant length:

1. fatha /a/,
2. kasra /i/,
3. damma /u/,
4. fathatayn (word-ending /an/),
5. kasratayn (word-ending /in/),
6. dammatayn (word-ending /un/),
7. shadda (consonant doubling), and
8. sukun (no vowel).

There are two approaches to handling this mismatch between the acoustics and transcripts. In the "unvowelized" approach, words are modeled graphemically, in terms of their letter sequences, and the acoustics corresponding to the unwritten diacritics are implicitly modeled by the Gaussian mixtures in the acoustic model. In the "vowelized" approach, words are modeled phonemically, in terms of their sound sequences, and the correct vowelization of transcribed words is inferred during training. Note that even when vowelized models are used the word error rate calculation is based on unvowelized references. Diacritics are typically not orthographically represented in Arabic texts. Diacritization is generally not necessary to make the transcript readable by Arabic literate readers. Thus, Arabic ASR systems typically do not output fully diacritized transcripts. Therefore, the vowelized forms are mapped back to unvowelized forms in scoring – it is also the NIST scoring scheme. In addition, the machine translation systems we use currently require unvowelized input. An excellent discussion of the Arabic language and automatic speech recognition can be found in Kirchhoff et al. [27].

One of the biggest challenges in building vowelized models is initialization: how to obtain a first set of vowelized models when only unvowelized transcripts are available. One approach is to have experts in Arabic manually vowelize a small training set [33]. The obvious disadvantage is that this process is quite labor intensive, which motivates researchers to explore automated methods [2]. Following the recipe in [2], we discuss our bootstrap procedure and some issues related to scaling up to large vocabularies.

Pronunciation Dictionaries

The words in the vocabulary of both the vowelized and unvowelized systems are assumed to be the same, and they do not contain any diacritics, just as they appear in most written text. In the unvowelized system, the pronunciation of a word is modeled by the sequence of letters in the word. For example, there is a single unvowelized pronunciation of the word *Abwh*.

Abwh(01) A b w h

The short vowels are not explicitly modeled, and it is assumed that speech associated with the short vowels will be implicitly modeled by the adjacent phones. In other words, short vowels are not presented in our phoneme set; acoustically, they will be modeled as part of the surrounding consonant acoustic models.

In the vowelized system, however, short vowels are explicitly modeled in both training and decoding. We use the Buckwalter Morphological Analyzer (Version 2.0) [7], and the Arabic Treebank to generate vowelized variants of each word. The pronunciation of each variant is modeled as the sequence of letters in the diacriticized word, including the short vowels. For *shadda* (consonant doubling), an additional consonant is added, and for *sukun* (no vowel), nothing is added. For example, there are four vowelized pronunciations of the written word *Abwh*.

Abwh(deny/refuse/+they+it/him)	A a b a w o h u
Abwh(desire/aspire/+they+it/him)	A a b b u w h u
Abwh(father+its/it)	A a b u w h u
Abwh(reluctant/unwilling+his/its)	A b u w h u

Table 13.1 Comparison of different initialization methods for vowelized models

Training method	WER (RT-04)
Flat-start	23.0 %
Bootstrap	22.8 %

The vowelized training dictionary has 243,368 vowelized pronunciations, covering a word list of 64,496 words. The vowelization rate is about 95 %. For the remaining 5 % of words that are not covered, we back off to unvowelized forms.

In the following subsections, we focus on the vowelized system. Since written transcripts of audio data do not usually contain short vowel information, how does one train the initial acoustic model? One could use a small amount of data with manually vowelized transcripts to bootstrap the acoustic model. Alternatively, one could perform flat-start training.

Flat-Start Training vs. Manual Transcripts

Our flat-start training procedure initializes context-independent HMMs by distributing the data equally across the HMM state sequence. We start with one Gaussian per state, and increase the number of parameters using mixture splitting interleaved within 30 forward/backward iterations. Now, the problem is that we have 3.8 vowelized pronunciations per word on average, but distributing the data requires a linear state graph for the initialization step. To overcome this problem, in the first iteration of training we randomly select pronunciation variants. All subsequent training iterations operate on the full state graph representing all possible pronunciations.

We compare this approach to manually vowelized transcripts where the correct pronunciation variant is given. BBN distributed 10 h of manually vowelized development data (BNAD-05, BNAT-05) that we used to bootstrap vowelized models. These models are then used to compute alignments for the standard training set (FBIS + TDT4). A new system is then built using fixed alignment training, followed by a few forward/backward iterations to refine the models. The error rates in Table 13.1 suggest that manually vowelized transcripts are not necessary. The fully automatic procedure is only 0.2 % worse. We opted for the fully automatic procedure in all our experiments, including the evaluation system.

Short Models for Short Vowels

We noticed that the vowelized system performed poorly on broadcast conversational speech. It appeared that the speaking rate is much faster, and that the vowelized state graph is too large to be traversed with the available speech frames. The acoustic models do not permit state skipping. One solution is to model the three short vowels with a shorter, 2-state HMM topology. The results are shown in Table 13.2.

Table 13.2 Comparison of different HMM topologies for short vowels

Topology	RT-04	BCAD-05
3-state	19.0 %	28.9 %
2-state	18.5 %	27.4 %

Table 13.3 OOV/WER ratio for an unvowelized system on RT-04

Vocabulary	OOV rate	WER
129k	2.9 %	20.3 %
589k	0.8 %	19.0 %

Table 13.4 OOV/WER ratio for a vowelized system on RT-04

Vocab.	Variants	Vowel. rate	OOV rate	WER
129k	538k	90.4 %	2.9 %	19.8 %
589k	1967k	72.6 %	0.8 %	18.3 %

The improvements on RT-04 (broadcast news) are relatively small; however, there is a 1.5 % absolute improvement on BCAD-05 (broadcast conversations).

Vowelization Coverage of the Test Vocabulary

As mentioned before, we back off to unvowelized forms for those words not covered by Buckwalter and Arabic Treebank. The coverage for the training dictionary is pretty high at 95 %. On the other hand, for a test vocabulary of 589k words, the vowelization rate is only 72.6 %. The question is whether it is necessary to manually vowelize the missing words, or whether we can get around that by backing off to the unvowelized pronunciations. One way to test this – without actually providing vowelized forms for the missing words – is to look at the OOV/WER ratio. The assumption is that the ratio is the same for a vowelized and an unvowelized system if the dictionary of the vowelized system does not pose any problems. More precisely, if we increase the vocabulary and we get the same error reduction for the vowelized system, then, most likely, there is no fundamental problem with the vowelized pronunciation dictionary.

For the unvowelized system, when increasing the vocabulary from 129k to 589k, we reduce the OOV rate from 2.9 % to 0.8 %, and we reduce the error rate by 1.3 % (Table 13.3). For the vowelized system, we see a similar error reduction of 1.5 % for the same vocabulary increase (Table 13.4). The system has almost 2 million vowelized pronunciations for a vocabulary of 589k words. The vowelization rate is about 72.6 %. In other words, 17.4 % of our list of 589k words are unvowelized in our dictionary. Under the assumption that we can expect the same OOV/WER ratio for both the vowelized and unvowelized system, the results in Table 13.3 and Table 13.4 suggest that the back-off strategy to the unvowelized forms is valid for our vowelized system.

Table 13.5 Effect of pronunciation probabilities on WER

System	RT-04	BCAD-05
Unvowelized	17.0 %	25.4 %
Vowelized	16.0 %	26.0 %
+ Pron. prob	14.9 %	25.1 %

Table 13.6 Comparison of vowelized and unvowelized models at various adaptation passes, no pronunciation probabilities. RT-04 test set

Decoding pass	Unvowelized	Vowelized
SI	29.5 %	25.0 %
VTLN	26.2 %	23.1 %
FMLLR	23.0 %	21.1 %
MLLR	22.1 %	20.4 %

Pronunciation Probabilities

Our decoding dictionary has about 3.3 pronunciations per word on average. Therefore, estimating pronunciation probabilities is essential to improve discrimination between the vowelized forms. We estimated the pronunciation probabilities by counting the variants in the 2,330-h training set.

The setup consists of ML models, and includes all the adaptation steps (VTLN, FMLLR, MLLR). The test sets are RT-04 (Broadcast News) and BCAD-05 (Broadcast Conversations). Adding pronunciation probabilities gives consistent improvements between 0.9 % and 1.1 % on all test sets (Table 13.5). Also, pronunciation probabilities are crucial for vowelized models; they almost double the error reduction from vowelization. We investigated several smoothing techniques and longer word contexts, but did not see any further improvements over simple unigram pronunciation probabilities.

Vowelization, Adaptation, and Discriminative Training

In this section, we summarize additional experiments with vowelized models. One interesting observation is that the gain from vowelization decreases significantly when more refined adaptation and training techniques are used. Table 13.6 shows the error rate of unvowelized and vowelized models with and without adaptation. The relative improvement from vowelization is more than 15 % at the speaker-independent level. However, after applying several normalization and adaptation techniques, the gain drops to 7.7 % at the MLLR level.

An even more drastic reduction of the vowelization gain is observed after discriminative training (Table 13.7). The vowelized setup includes the 2-state HMMs for short vowels and pronunciation probabilities. While discriminative training reduces the error rate by 4.9 % for the unvowelized setup, we observed an error reduction of only 3.4 % for the vowelized models.

It seems that standard adaptation and discriminative techniques can at least partially compensate for the invalid model assumption of ignoring short vowels, and the improvements from vowelization are subsequently reduced when using

Table 13.7 Comparison of vowelized and unvowelized models before and after discriminative training. Vowelized system uses pronunciation probabilities and 2-state HMMs for short vowels. Both systems use speaker adaptation: VTLN, FMLLR, and MLLR. DEV-07 test set

Models	Unvowelized	Vowelized
ML	17.1 %	14.8 %
fMPE + MPE	12.2 %	11.4 %

Table 13.8 Acoustic models trained without and with additional TRANSTAC data

Acoustic model	WER (DEV-07)
2330h GALE	19.2 %
+ 500h TRANSTAC	19.6 %

better adaptation and training techniques. Thus, well-engineered speech recognition systems using only language-independent techniques can perform almost as well as systems using language-specific methods.

13.2.3 Modeling of Arabic Dialects in Decision Trees

As shown in Table 13.28, the training data comes from a variety of sources, and there are systematic variations in the data, including dialects (Modern Standard Arabic, Egyptian Arabic, Gulf Arabic, etc.), broadcasters (Al Jazeera, Al Arabiya, LBC, etc.), and programs (Al Jazeera morning news, Al Jazeera news bulletin, etc.).

The problem we face is how to build acoustic models on diverse training data. Simply adding more training data from a variety of sources does not always improve system performance. To demonstrate this, we built two acoustic models with identical configurations. In one case, the model was trained on GALE data (2,330 h, including unsupervised BN-03), while in the second case we added 500 h of TRANSTAC data to the training set. TRANSTAC data contains Iraqi Arabic (Nadji spoken Arabic and Mesopotamian Arabic). Both GALE and TRANSTAC data are Arabic data, but they represent very different dialects and styles. The model trained on additional TRANSTAC data is 0.4 % worse than our GALE model (see Table 13.8).

Both acoustic models used 400k Gaussians, a comparatively large number that should be able to capture the variability of different data sources. Furthermore, we did not find that increasing the number of Gaussians could offset the reduced performance caused by the addition of unmatched training data.

Because adding more training data will not always improve performance, we need to find which part of the training data is relevant. Training separate models for each category or dialect requires sufficient training data for each category, significantly more human and computational effort, and algorithms that reliably cluster training and test data into meaningful categories. We propose instead to model dialects directly in the decision trees: a data-driven method for building dialect-specific models without splitting the data into different categories.

Similiar approaches using additional information in decision trees can be found in the literature. Reichl and Chou [37] used gender information to specialize decision trees for acoustic modeling. Speaking rate, SNR, and dialects were used in [19], and questions about the speaking mode were used in [45] to build models for hyperarticulated speech.

Decision Trees with Dialect Questions

We extend our regular decision tree algorithm by including non-phonetic questions about dialects. The question set contains our normal phonetic context questions, as well as dynamic questions regarding dialect. Dialect questions compete with phonetic context questions to split nodes in the tree. If dialect information is irrelevant for some phones, it will simply be pruned away during tree training.

The training of dialect-specific trees and Gaussian mixture models is straightforward. The decision tree is grown in a top-down clustering procedure. At each node, all valid questions are evaluated, and the question with the best increase in likelihood is selected. We use single Gaussians with diagonal covariances as node models. Statistics for each unclustered context-dependent HMM state are generated in one pass over the training data prior to the tree growing. Dialect information is added during HMM construction by tagging phones with additional information. The additional storage cost this entails is quite significant: the number of unique, unclustered context-dependent HMM states increases roughly in proportion to the number of different tags used.

The questions used for tree clustering are written as conjugate normal forms. Literals are basic questions about the phone class or tag for a given position. This allows for more complex questions such as *Is the left context a vowel and is the channel Al Jazeera ($-1 = Vowel$ && $0 = AlJazeera$).* Similarly, more complex questions on the source may be composed. For example, to ask for Al Jazeera, one would write *($0 = AlJazeeraMorning$ or $0 = AlJazeeraAfternoon$).* The idea is to allow a broad range of possible questions, and to let the clustering procedure select the relevant questions based on the training data. The questions cover all the channel and dialect information available from the audio filenames.

We used this technique to build a dialect-dependent decision tree for the acoustic models trained on the combined GALE and TRANSTAC data. We generated a tree with 15,865 nodes, and 8,000 HMM states. Approximately 44 % of the states depend on the dialect tag, so the remaining 56 % of the models are shared between two very different data sources: GALE and TRANSTAC.

Building Static Decoding Graphs for Dynamic Trees

Training dialect-specific models is relatively easy; however, decoding with such models is more complicated if a statically compiled decoding graph is used. The problem is that the decision tree contains dynamic questions that can be answered

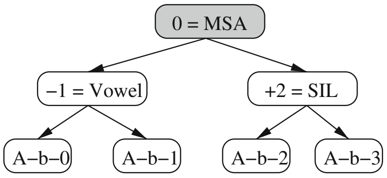

Fig. 13.5 Decision tree with dialect questions. Each question is of the format *ContextPosition* = *Class*. Leaves are marked with HMM states *A-b-0,...* (phone */A/*, begin state)

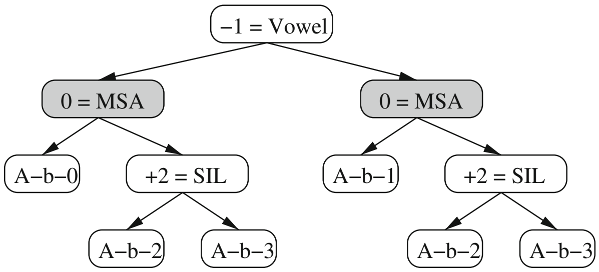

Fig. 13.6 Decision tree after one transformation step. The original root *0* = *MSA* was replaced by the left child −*1* = *Vowel*, and a new copy of *0* = *MSA* was created

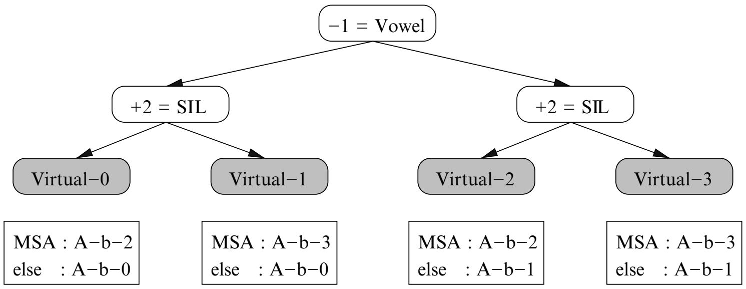

Fig. 13.7 Reordered decision tree. Dynamic questions are replaced by virtual leaves

only at run-time, and not when the graph is compiled. The solution for this problem is to separate the decision tree into two parts: a static part containing only phonetic questions, and a dynamic part for the dialect questions. The decision tree is reordered such that no dynamic question occurs above a static question. The static part of the decision tree can be compiled into a decoding graph as usual, while the dynamic part of the tree is replaced by a set of virtual leaves. The decoder maintains a lookup table that transforms each virtual leaf to a corresponding dialect-specific leaf at run-time.

In the following we explain how to reorder the decision tree such that dynamic questions do not occur above static questions. Figures 13.5–13.7 illustrate the process. In this example, the root of the tree is marked with the question *0* = *MSA*.

Table 13.9 Reordered tree statistics for GALE + TRANSTAC setup

Number of nodes	15,865
Number of leaves	8,000
Dialect-dependent leaves	44.3 %
Number of reordered nodes	29,137
Number of virtual leaves	7,085

If the center phone is marked as Modern Standard Arabic (MSA), the right branch will be chosen, otherwise the left branch.

The reordering algorithm consists of a sequence of transformation steps. In each step, a dynamic question node is pushed down one level by replacing it with one of its children nodes. Figure 13.6 shows the tree after applying one transformation step. In each transformation step, the selected dynamic question node is duplicated together with one of its branches. In this example, the right branch starting with $+2 = SIL$ is duplicated. The node to be moved up (promoted) ($-1 = Vowel$ in Fig. 13.6) is separated from its subtrees and moved up one level, with the duplicated subtrees becoming its children. The subtrees that were originally children of the promoted node become its grandchildren. The resulting tree is equivalent to the original tree, but the dynamic question is now one level deeper.

The reordering procedure terminates when no transformation step can be applied. In the worst case, this procedure causes the decision tree's size to grow exponentially; however, in practice we observe only moderate growth. Table 13.9 summarizes the tree statistics after reordering. The number of internal nodes grew by only a factor of 2, which is easily managed for decoding graph construction.

The last step is to separate the static and dynamic tree parts. The dynamic question nodes are replaced by virtual leaves for the graph construction. The virtual leaves correspond to lookup tables that map virtual leaves to physical HMM states at run-time. The static decoding graph can now be constructed using the tree with virtual leaves. At run-time, dialect information is available,[1] and virtual leaves can be mapped to the corresponding physical HMM states for acoustic score computation.

Experiments

We use our vowelized Arabic model as a baseline in our experiments. The vocabulary has about 737,000 words, and 2.5 million pronunciations. The language model is a 4-gram LM with 55M n-grams. Speaker adaptation includes VTLN and FMLLR. All models are ML trained. In addition to the GALE EVAL-06 and DEV-07 test sets, we also used a TRANSTAC test set comprising 2 h of audio.

[1]This can be done via a separate dialect ID tool, as described in Sect. 13.5.1, selecting the dialect with the best likelihood, or other sources of information.

Table 13.10 Comparison of regular tree and tree with dialect questions. GALE DEV-07 test set

Acoustic model	Regular tree	Dialect tree
2330h GALE	19.2 %	18.6 %
+ 500h TRANSTAC	19.6 %	18.7 %

Table 13.11 Comparison of regular tree and tree with dialect questions. TRANSTAC test set

Acoustic model	Regular tree	Dialect tree
2330h GALE	35.9 %	–
+ 500h TRANSTAC	25.9 %	24.7 %

Table 13.12 Comparison of regular tree and tree with dialect questions with unsupervised training data. GALE EVAL-06 and DEV-07 test sets

Test set	Regular tree	Dialect tree
EVAL-06	30.2 %	29.6 %
DEV-07	20.3 %	19.5 %

The dialect labels are derived from the audio file names. The file names encode TV channel and program information.

In the first experiment we train four acoustic models. Each model has 8,000 states and 400,000 Gaussians. The models are trained on either 2,330 h of GALE data or on the GALE data plus 500 h of TRANSTAC data. For each training set, we train one model using the regular (phonetic context only) question set and one using phonetic and dialect questions. The test set is DEV-07. The results are summarized in Table 13.10. For the GALE model, we see an improvement of 0.6 % WER. The improvement for the GALE + TRANSTAC training set is slightly higher, 0.9 %. The results suggest that the decision tree with dialect questions can better cope with diverse, and potentially mismatched, training data.

In the second experiment (Table 13.11), we use the same set of acoustic models as before, but the vocabulary and language model are now TRANSTAC-specific and the test set is drawn from TRANSTAC data. Adding TRANSTAC data improves the error rate from 35.9 to 25.9 %. Adding the dialect-specific questions to the tree-building process improves the error rate by an additional 1.2 %. We did not test the dialect tree trained on GALE data only. The tree does not contain any TRANSTAC-related questions, since the models were trained on GALE data only.

In the final experiment, we use a large amount of unsupervised training data from our internal TALES data collection. The acoustic model was trained on 2,330 h of GALE data plus 5,600 h of unsupervised training data. The results are shown in Table 13.12. The dialect-dependent decision tree reduces the error rate by 0.6 % on EVAL-06 and 0.8 % on DEV-07. While adding more unsupervised training data does not help if large amounts of supervised training data are available, we observe that the dialect tree is able to compensate for adding "unuseful" data (Table 13.12).

13.3 Language Modeling

13.3.1 Language-Independent Techniques for Language Modeling

Base N-Grams

The standard speech recognition model for word sequences is the n-gram language model. To see how an n-gram model is derived, consider expanding the joint probability of a sequence of M words in terms of word probabilities conditioned on word histories:

$$P(w_M, w_{M-1}, \ldots, w_3, w_2, w_1) = P(w_1) \times P(w_2|w_1) \times P(w_3|w_2, w_1) \times \cdots \times P(w_M|w_{M-1}, \ldots, w_3, w_2, w_1) \tag{13.4}$$

Given that the words are drawn from a dictionary containing tens to hundreds of thousands of words, it is clear that the probability of observing a given sequence of words becomes vanishingly small as the length of the sequence increases. The solution is to make a Markov assumption that the probability of a word is conditionally independent of previous words, given a history of fixed length h. That is,

$$P(w_M, w_{M-1}, \ldots, w_3, w_2, w_1) \approxeq P(w_1) \times P(w_2|w_1) \times P(w_3|w_2, w_1) \times \cdots \times P(w_M|w_{M-1}, w_{M-2}) \tag{13.5}$$

A model that makes a first-order Markov assumption, conditioning words only on their immediate predecessors, is called a *bigram* model, because it deals with pairs of words. Likewise, a model that makes a second-order Markov assumption is called a *trigram* model because it deals with word triplets: two-word histories and the predicted word. Equation (13.5) illustrates a trigram model.

N-gram language models are trained by collecting a large amount of text and simply counting the number of times each word occurs with each history. However, even with the n-gram assumption, there is still a problem with data sparsity: a model based only on counting the occurrences of words and histories in some training corpus will assign zero probability to legal word sequences that the speech recognition system should be able to produce. To cope with this, various forms of smoothing are used on language models that reassign some probability mass from observed events to unobserved events. The models described below generally use modified Kneser–Ney smoothing [1, 13].

Model M

Model M [11] is a class-based n-gram model; its basic form is as follows:

$$p(w_1 \cdots w_l) = \prod_{j=1}^{l+1} p(c_j | c_{j-2} c_{j-1}, w_{j-2} w_{j-1}) p(w_j | w_{j-2} w_{j-1} c_j) \tag{13.6}$$

It is composed of two submodels, a model predicting classes and a model predicting words, both of which are exponential models. An exponential model $p_\Lambda(y|x)$ is a model with a set of *features* $\{f_i(x, y)\}$ and equal number of parameters $\Lambda = \{\lambda_i\}$ where

$$p_\Lambda(y|x) = \frac{\exp(\sum_i \lambda_i f_i(x, y))}{Z_\Lambda(x)} \tag{13.7}$$

and where $Z_\Lambda(x)$ is a normalization factor defined as

$$Z_\Lambda(x) = \sum_{y'} \exp(\sum_{i=1}^{F} \lambda_i f_i(x, y')) \tag{13.8}$$

Let $p_{\text{ng}}(y|\omega)$ denote an exponential n-gram model, where we have a feature $f_{\omega'}$ for each suffix ω' of each ωy occurring in the training set; this feature has the value 1 if ω' occurs in the current event and 0 otherwise. For example, the model $p_{\text{ng}}(w_j | w_{j-1} c_j)$ has a feature f_ω for each n-gram ω in the training set of the form w_j, $c_j w_j$, or $w_{j-1} c_j w_j$. Let $p_{\text{ng}}(y|\omega_1, \omega_2)$ denote a model containing all features in $p_{\text{ng}}(y|\omega_1)$ and $p_{\text{ng}}(y|\omega_2)$. Then, the distributions in Eq. (13.6) are defined as follows for the trigram version of Model M:

$$p(c_j | c_{j-2} c_{j-1}, w_{j-2} w_{j-1}) \equiv p_{\text{ng}}(c_j | c_{j-2} c_{j-1}, w_{j-2} w_{j-1}) \tag{13.9}$$

$$p(w_j | w_{j-2} w_{j-1} c_j) \equiv p_{\text{ng}}(w_j | w_{j-2} w_{j-1} c_j) \tag{13.10}$$

To smooth or regularize Model M, it has been found that $\ell_1 + \ell_2^2$ regularization works well; i.e., the parameters $\Lambda = \{\lambda_i\}$ of the model are chosen to minimize

$$\mathscr{O}_{\ell_1 + \ell_2^2}(\Lambda) = \log \text{PP}_{\text{train}} + \frac{\alpha}{C_{\text{tot}}} \sum_i |\lambda_i| + \frac{1}{2\sigma^2 C_{\text{tot}}} \sum_i \lambda_i^2 \tag{13.11}$$

where PP_{train} denotes training set perplexity and where C_{tot} is the number of words in the training data. The values α and σ are regularization hyperparameters, and the values ($\alpha = 0.5, \sigma^2 = 6$) have been found to give good performance for a wide range of operating points. A variant of iterative scaling can be used to find the parameter values that optimize Eq. (13.11).

Neural Network Language Model

The neural network language model (NNLM) [3, 17, 44] uses a continuous representation of words, combined with a neural network for probability estimation. The model size increases linearly with the number of context features, in contrast to exponential growth for regular n-gram models. Details of our implementation, speed-up techniques, as well as the probability normalization and optimal NN configuration, are described in [18].

The basic idea behind neural network language modeling is to project words into a continuous space and let a neural network learn the prediction in that continuous space, where the model estimation task is presumably easier than the original discrete space [3, 17, 44]. The continuous space projections, or *feature vectors*, of the preceding words (or context features) make up the input to the neural network, which then will produce a probability distribution over a given vocabulary. The feature vectors are randomly initialized and are subsequently learned, along with the parameters of the neural network, so as to maximize the likelihood of the training data. The model achieves generalization by assigning to an unseen word sequence a probability close to that of a "similar" word string seen in the training data. The similarity is defined as being close in the multi-dimensional feature space. Since the probability function is a smooth function of the feature vectors, a small change in the features leads to only a small change in the probability.

To compute the conditional probability $P(y|x_1, x_2, \cdots, x_m)$, where $x_i \in V_i$ (*input vocabulary*) and $y \in V_o$ (*output vocabulary*), the model operates as follows: First for every $x_i, i = 1, \cdots, m$, the corresponding feature vector (continuous space projection) is found. This is simply a table lookup operation that associates a real vector of fixed dimension d with each x_i. Secondly, these m vectors are concatenated to form a vector of size $m \cdot d$. Finally this vector is processed by the neural network which produces a probability distribution $P(.|x_1, x_2, \cdots, x_m)$ over vocabulary V_o at its output.

Note that the input and output vocabularies V_i and V_o are independent of each other and can be completely different. Training is achieved by searching for parameters Φ of the neural network and the values of feature vectors that maximize the penalized log-likelihood of the training corpus:

$$L = \frac{1}{T} \sum_t logP(y^t|x_1^t, \ldots, x_m^t; \Phi) - R(\Phi) \tag{13.12}$$

where superscript t denotes the tth event in the training data, T is the training data size and $R(\Phi)$ is a regularization term, which in our case is a factor of the L2 norm squared of the hidden and output layer weights.

The model architecture is given in Fig. 13.8 [3,17,44]. The neural network is fully connected and contains one hidden layer. The operations of the input and hidden layers are given by:

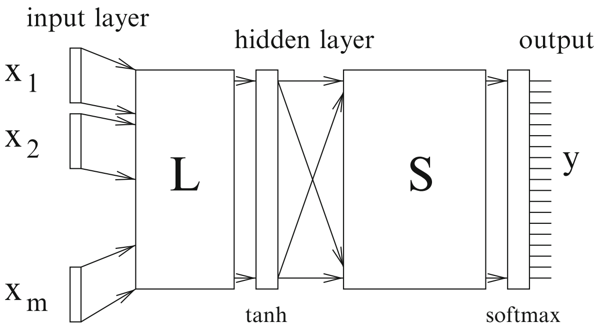

Fig. 13.8 The neural network architecture

$$\mathbf{f} = (f_1, \ldots, f_{d \cdot m}) = (\mathbf{f}(x_1), \mathbf{f}(x_2), \cdots, \mathbf{f}(x_m))$$
$$g_k = \tanh\left(\textstyle\sum_j f_j L_{kj} + B_k^1\right) \qquad k = 1, 2, \ldots, h$$

where $\mathbf{f}(x)$ is the d-dimensional feature vector for token x. The weights and biases of the hidden layer are denoted by L_{kj} and B_k^1 respectively, and h is the number of hidden units.

At the output layer of the network we have:

$$z_k = \textstyle\sum_j g_j S_{kj} + B_k^2 \qquad k = 1, 2, \ldots, |V_o|$$

$$p_k = \frac{e^{z_k}}{\sum_j e^{z_j}} \qquad k = 1, 2, \ldots, |V_o| \tag{13.13}$$

with the weights and biases of the output layer denoted by S_{kj} and B_k^2 respectively. The softmax layer (Eq. (13.13)) ensures that the outputs are valid probabilities and provides a suitable framework for learning a probability distribution.

The kth output of the neural network, corresponding to the kth item y_k of the output vocabulary, is the desired conditional probability: $p_k = P(y^t = y_k | x_1^t, \ldots, x_m^t)$.

The neural network weights and biases, as well as the input feature vectors, are learned simultaneously using stochastic gradient descent training via back-propagation algorithm, with the objective function being the one given in Eq. (13.12). Details of the implementation, speed-up techniques, as well as the probability normalization and optimal NN configuration, are described in [18].

Given the large vocabulary of the Arabic speech recognition system, data sparsity is an important problem for conventional n-gram LMs. Our experience is that NNLM significantly reduces perplexity as well as word error rate in speech recognition. Results will be presented later together with those of an NNLM that incorporates syntactic features.

13.3.2 Language-Specific Techniques for Language Modeling

In this section, we describe a language model that incorporates morphological and syntactic features for Arabic speech recognition [29, 30]. This method is language specific in the sense that certain language-specific resources are used, for example, an Arabic parser.

With a conventional n-gram language model, the number of parameters can potentially grow exponentially with the context length. Given a fixed training set, as n is increased, the number of unique n-grams that can be reliably estimated is reduced. Hence in practice, a context no longer than three words (corresponding to a 4-gram LM) is used.

This problem is exacerbated by large vocabularies for morphologically rich languages like Arabic. Whereas the vocabulary of an English speech recognition system typically has under 100k words, our Arabic system has about 800k words. The idea of using rich morphology information in Arabic language modeling has been explored by several researchers. The most common idea has been to use segments, which are the result of breaking an inflected word into parts, for better generalization when estimating the probabilities of n-gram events [28]. As an example, the white-space delimited word *tqAblhm* (she met them) is segmented into three morphs: prefix *t* (she), followed by stem *qAbl* (met) and suffix *hm* (them).

Compared to a regular word n-gram model, a segmented word n-gram model has a reduced context. To model longer-span dependencies, one may consider context features extracted from a syntactic parse tree such as head word information used in the Structured Language Model (SLM) [10]. Syntactic features can be useful in any language. Here is an example in English:

> The *girl* who lives down the street *searched* the bushes in our neighbor's backyard *for* her lost kitten.

Through a parse tree, one can relate the words *girl*, *searched*, and *for*. Such long-span relationships cannot be captured with a 4-gram language model. Various types of syntactic features such as head words, non-terminal labels, and part-of-speech tags, have been used in a discriminative LM framework as well as in other types of models [15].

Using many types of context features (morphological and syntactic) makes it difficult to model with traditional back-off methods. Learning the dependencies in such a long context is difficult even with models such as factored language models [28] due to the large number of links that need to be explored. On the other hand, the neural network language model (NNLM) [3, 17, 44] is very suitable for modeling such context features. The NNLM uses a continuous representation of words, combined with a neural network for probability estimation. The model size increases linearly with the number of context features, in contrast to exponential growth for regular n-gram models. Another advantage is that it is not required to define a back-off order of the context features. The model converges to the same solution no matter how the context features are ordered, as long as the ordering is consistent.

Table 13.13 WER results of NNLM using word and syntactic features

LM	EVAL08U	BN	BC
4-gram	9.4 %	6.9 %	12.5 %
4-gram + word NNLM	9.1 %	6.5 %	12.1 %
4-gram + word NNLM + syntax NNLM	8.6 %	6.2 %	11.5 %

The following text processing steps are used to extract morphological and syntactic features for context modeling with an NNLM. Arabic sentences are processed to segment words into (hypothesized) prefixes, stems, and suffixes, which become the tokens for further processing. In particular, we use Arabic Treebank (ATB) segmentation, which is a *light* segmentation adopted to the task of manually writing parse trees in the ATB corpus [32]. After segmentation, each sentence is parsed, and syntactic features are extracted. The context features used by the NNLM include segmented words (morphs) as well as syntactic features such as exposed head words and their non-terminal labels.

Table 13.13 shows the WER results of using NNLMs, with word features only or with morphological and syntactic features. The results are given for an evaluation set EVAL08U, with a breakdown for the broadcast news (BN) and broadcast conversations (BC) portions. An NNLM with word features reduced the WER by about 3 % relative, from 9.4 to 9.1 %. Using morphological and syntactic features further reduced the WER by 5 % relative, from 9.1 to 8.6 %. It is seen that syntactic features are helping both BN and BC. Specifically, through the use of syntactic features, for BN, the WER improved by 4.6 % (6.5–6.2 %) and for BC, the WER improved by 5.0 % (12.1–11.5 %).

Although the modeling methodology behind the syntax NNLM is language independent, when a new language or dialect is encountered, certain resources such as a segmenter and parser may have to be developed or adapted. In our experiments, even though the syntax NNLM was trained on only 12 million words of data, two orders of magnitude less than the text corpora of over 1 billion words used to train the n-gram LM, it was still able to provide significant improvements to the WER. For a new language with little data available to train the n-gram LM, the syntax LM is likely to help even more.

Search

The search space for large-vocabulary speech recognition is both enormous and complex. It is enormous because the number of hypotheses to consider grows exponentially with the length of the hypothesized word strings and because the number of different words that can be recognized is typically in the tens to hundreds of thousands. The search space is complex because there is significant structure induced by the language model, the dictionary, and the decision trees used for HMM state clustering. One approach to search, the *static* approach, represents the components of the search space (the language model, dictionary, and decision trees) as weighted finite-state transducers (WFSTs), and then uses standard algorithms such as WFST composition, determinization, and minimization to pre-compile a *decoding graph* that represents the search space. The search process then simply reads in this graph and performs dynamic programming search on it, given a sequence of acoustic feature vectors, to perform speech recognition. Such static decoders can be very efficient and can be implemented with very little code because all of the complexity is pushed into the precompilation process. However, the size of the language models that can be used with static decoders is limited by what models can successfully be precompiled into decoding graphs. An alternative approach, *dynamic search*, constructs the relevant portion of the search space on the fly. Such decoders can use much larger language models, but are also significantly more complicated to write.

13.4 IBM GALE 2011 System Description

In this section we describe IBM's 2011 transcription system for Arabic broadcasts, which was fielded in the GALE Phase 5 machine translation evaluation. Like most systems fielded in competitive evaluations, our system relies upon multiple passes of decoding, acoustic model adaptation, language model rescoring, and system combination to achieve the lowest possible word error rate.

13.4.1 Acoustic Models

We use an acoustic training set composed of approximately 1,800 h of transcribed Arabic broadcasts provided by the Linguistic Data Consortium (LDC) for the GALE evaluations.

Unless otherwise specified, all our acoustic models use 40-dimensional features that are computed by an LDA projection of a supervector composed from 9 successive frames of 13-dimensional mean- and variance-normalized PLP features followed by diagonalization using a global semi-tied covariance transform [20], and use pentaphone cross-word context with a "virtual" word-boundary phone symbol that occupies a position in the context description, but does not generate an acoustic observation. Speaker-adapted systems are trained using VTLN and fMLLR. All the models use variable frame rate processing [14].

Given that the short vowels and other diacritic markers are typically not orthographically represented in Arabic texts, we have a number of choices for building pronunciation dictionaries: (1) unvowelized (graphemic) dictionaries in which the short vowels and diacritics are ignored, (2) vowelized dictionaries which use the Buckwalter morphological analyzer [7] for generating possible vowelized pronunciations and (3) vowelized dictionary which uses the output of a morphological analysis and disambiguation tool (MADA) [23]; the assignment of such diacritic markers is based on the textual context of each word (to distinguish word senses and grammatical functions).[2] Our 2011 transcription system uses the acoustic models described below.

- **SI** A speaker-independent, unvowelized acoustic model trained using model-space boosted maximum mutual information [35]. The PLP features for this system are only mean-normalized. The **SI** model comprises 3k states and 151k Gaussians.
- **U** A speaker-adapted, unvowelized acoustic model trained using both feature- and model-space BMMI. The **U** model comprises 5k states and 803k Gaussians.
- **SGMM** A speaker-adapted, Buckwalter vowelized subspace Gaussian mixture model [36,43] trained with feature- and model-space versions of a discriminative criterion based on both the minimum phone error (MPE) [34] and BMMI criteria. The **SGMM** model comprises 6k states and 150M Gaussians that are represented using an efficient subspace tying scheme.
- **V** A speaker-adapted, Buckwalter vowelized acoustic model trained using the feature-space BMMI and model-space MPE criteria. The changes in this model compared to all the other models are: (1) the "virtual" word boundary phones are replaced with word-begin and word-end tags, (2) it uses a dual decision tree that specifies 10k different Gaussian mixture models, but 50k context-dependent states, (3) it uses a single, global decision tree and (4) expands the number of phones on which a state can be conditioned to ± 3 within words. This model has 801k Gaussians.
- **BS** A speaker-adapted, unvowelized acoustic model using Bayesian sensing HMMs where the acoustic feature vectors are modeled by a set of state-dependent basis vectors and by time-dependent sensing weights [40]. The Bayesian formulation comes from assuming state-dependent Gaussian priors for the weights and from using marginal likelihood functions obtained by integrating out the weights. The marginal likelihood is Gaussian with a factor analyzed covariance matrix with the basis providing a low-rank correction to the diagonal covariance of the reconstruction error [42]. The details of this model are given in Sect. 13.4.1.
- **M** A speaker-adapted system, MADA vowelized system, with an architecture similar to **V**. The details of this model are given in Sect. 13.4.1.
- **NNU, NNM** Speaker-adapted acoustic models which use neural network features. They were built using either the unvowelized lexicon (**NNU**) or the MADA one (**NNM**). Section 13.4.1 describes these models in more detail.

[2]MADA operates by examining a list of all possible Buckwalter morphological analyses for each word, and then selecting the analysis that matches the current context best using SVM classifiers. MADA uses 19 distinct weighted morphological features. The selected analyses carry complete diacritic, lexemic, glossary and morphological information; thus all disambiguation decisions are made in one step. See [23] for more details.

Bayesian Sensing HMMs (BS)

Model Description

Here, we briefly describe the main concepts behind Bayesian sensing hidden Markov models [40]. The state-dependent generative model for the D-dimensional acoustic feature vectors $\mathbf{x}_t$ is assumed to be

$$\mathbf{x}_t = \Phi_i \mathbf{w}_t + \boldsymbol{\epsilon}_t \tag{13.14}$$

where $\Phi_i = [\boldsymbol{\phi}_{i1}, \ldots, \boldsymbol{\phi}_{iN}]$ is the basis (or dictionary) for state i and $\mathbf{w}_t = [w_{t1}, \ldots, w_{tN}]^T$ is a time-dependent weight vector. The following additional assumptions are made: (1) when conditioned on state i, the reconstruction error is zero-mean Gaussian distributed with precision matrix R_i, i.e. $\boldsymbol{\epsilon}_t | s_t = i \sim \mathcal{N}(\mathbf{0}, R_i^{-1})$ and (2) the state-conditional prior for $\mathbf{w}_t$ is also zero-mean Gaussian with precision matrix A_i, that is $\mathbf{w}_t | s_t = i \sim \mathcal{N}(\mathbf{0}, A_i^{-1})$. It can be shown that, under these assumptions, the marginal state likelihood $p(\mathbf{x}_t | s_t = i)$ is also zero-mean Gaussian with the factor analyzed covariance matrix [42]

$$S_i \overset{\Delta}{=} R_i^{-1} + \Phi_i A_i^{-1} \Phi_i^T \tag{13.15}$$

In summary, the state-dependent distributions are fully characterized by the parameters $\{\Phi_i, R_i, A_i\}$. In [40], we discuss the estimation of these parameters according to a maximum likelihood type II criterion, whereas in [41] we derive parameter updates under a maximum mutual information objective function.

Automatic Relevance Determination

For diagonal $A_i = \text{diag}(\alpha_{i1}, \ldots, \alpha_{iN})$, the estimated precision matrix values α_{ij} encode the relevance of the basis vectors $\boldsymbol{\phi}_{ij}$ for the dictionary representation of $\mathbf{x}_t$. This means that one can use the trained α_{ij} for controlling model complexity. One can first train a large model and then prune it to a smaller size by discarding the basis vectors which correspond to the largest precision values of the sensing weights.

Initialization and Training

We first train a large acoustic model with 5,000 context-dependent HMM states and 2.8 million diagonal covariance Gaussians using maximum likelihood in a discriminative FMMI feature space. The means of the GMM for each state are then clustered using k-means. The initial bases are formed by the clustered means. The resulting number of mixture components for the Bayesian sensing models after the clustering step was 417k. The precision matrices for the sensing weights and the reconstruction errors are assumed to be diagonal and are initialized to the identity matrix.

The models are trained with six iterations of maximum likelihood type II estimation. Next, we discard 50 % of the basis vectors corresponding to the largest precision values of the sensing weights and retrain the pruned model for two additional ML type II iterations. We then generate numerator and denominator lattices with the pruned models and perform four iterations of boosted MMI training of the model parameters as described in [41]. The effect of pruning and discriminative training is discussed in more details in [42].

MADA-Based Acoustic Model (M)

This acoustic model is similar to the **V** model. It uses a global tree, word position tags, and a large phonetic context of ± 3 . While the MADA-based model uses approximately the same number of Gaussians, the decision tree uses only one level, keeping the number of HMM states to 10,000. Since the MADA-based model uses a smaller phone set than the Buckwalter vowelized models, we were able to reuse the vowelized alignments and avoid the flat-start procedure. In this section we describe the strategy used for constructing training and decoding pronunciation dictionaries, the main difference between this system and the **V** system. Both pronunciation dictionaries are generated following [5] with some slight modification.

Training Pronunciation Dictionary

Here we describe an automatic approach to building a pronunciation dictionary that covers all words in the orthographic transcripts of the training data. First, for each utterance transcript, we run MADA to disambiguate each word based on its context in the transcript. MADA outputs all possible fully-diacritized morphological analyses for each word, ranked by their confidence, the MADA confidence score. We thus obtain a fully-diacritized orthographic transcription for training. Second, we map the highest-ranked diacritization of each word to a set of pronunciations, which we obtain from the 15 pronunciation rules described in [5]. Since MADA may not always rank the best analysis as its top choice, we also run the pronunciation rules on the **second** best choice returned by MADA, when the difference between the top two choices is less than a threshold determined empirically (in our implementation we chose 0.2). The IBM system is flexible enough to allow specifying multiple diacritized word options at the (training) transcript level. A sentence can be a sequence of fully diacritized word pairs as opposed to a sequence of single words. This whole process gives us fully disambiguated and diacritized training transcripts with more than one or two options per word.

Decoding Pronunciation Dictionary

For building the decoding dictionary we run MADA on the transcripts of the speech training data as well as on the Arabic Gigaword corpus. In this dictionary, all pronunciations produced (by the pronunciation rules) for all diacritized word instances (from MADA first and second choices) of the same undiacritized form are mapped to the undiacritized and *normalized* word form. Word normalization here refers to removing diacritic markers and replace Buckwalter normalized Hamzat-Wasl ({), <, and > by the letter 'A'. Note that it is standard to produce undiacritized transcripts when recognizing MSA. Diacritization is generally not necessary to make the transcript readable by Arabic-literate readers. Therefore, entries in the decoding pronunciation dictionary need only to consist of undiacritized words mapped to a set of phonetically-represented diacritizations.

A pronunciation confidence score is calculated for each pronunciation. We compute a pronunciation score s for a pronunciation p as the average of the MADA confidence scores of the MADA analyses of the word instances that this pronunciation was generated from. We compute this score for each pronunciation of a normalized undiacritized word. Let m be the maximum of these scores. Now, the final pronunciation confidence score for p is $-log_{10}(c/m)$. This basically means that the best pronunciation receives a penalty of 0 when chosen by the ASR decoder. This dictionary has about 3.6 pronunciations per word when using the first and second MADA choices.

Neural Network Acoustic Models (NNU and NNM)

The neural network feature extraction module uses two feature streams computed from mean and variance normalized, VTLN log Mel spectrograms, and is trained in a piecewise manner, in which (1) a state posterior estimator is trained for each stream, (2) the unnormalized log-posteriors from all streams are summed together to combine the streams, and (3) features for recognition are computed from the bottleneck layer of an autoencoder network. One stream, the lowpass stream, is computed by filtering the spectrograms with a temporal lowpass filter, while the other stream, the bandpass stream, is computed by filtering the spectrograms with a temporal bandpass filter. Both filters are 19-point FIR filters. The lowpass filter has a cutoff frequency of 24 Hz. The bandpass filter has a differentiator-like (gain proportional to frequency) response from 0 to 16 Hz and a high-pass cutoff frequency of 27 Hz. The posterior estimators for each stream compute the probabilities of 141 context-independent HMM states given an acoustic input composed from 19 frames of 40-dimensional, filtered spectrograms. They have two 2048-unit hidden layers, use softsign nonlinearities [21] between layers, and use a softmax nonlinearity at the output. The softsign nonlinearity is $y = x/(1 + |x|)$. Initial training optimizes the frame-level cross-entropy criterion. After convergence, the estimators are further refined to discriminate between state sequences using the minimum phone error criterion [24, 34]. Stream combination is performed by discarding the softmax output layer for each stream posterior estimator, and summing the resulting outputs, which may be interpreted as unnormalized log-posterior probabilities. We then train another neural network, containing a 40-dimensional bottleneck layer, as an autoencoder, and use the trained network to reduce the dimensionality of the neural network features. The original autoencoder network has a first hidden layer of 76 units, a second hidden layer of 40 units, a linear output layer, and uses softsign nonlinearities. The training criterion for the autoencoder is the cross-entropy between the *normalized* posteriors generated by processing the autoencoder input and output vectors through a softmax nonlinearity. Once the autoencoder is trained, the second layer of softsign nonlinearities and the weights that expand from the 40-dimensional bottleneck layer back to the 141-dimensional output are removed. Details of this NN architecture are described in [39].

Once the features are computed, the remaining acoustic modeling steps are conventional, using 600k 40-dimensional Gaussians modeling 10k quinphone context-dependent states, where we do both feature- and model-space discriminative training using the BMMI criterion. Two acoustic models were trained using the neural-net features: one (**NNM**) used a MADA-vowelized lexicon, while the other (**NNU**) used an unvowelized lexicon. Note that the posterior estimators used in feature extraction were trained with MADA-vowelized alignments.

13.4.2 Language Models

For training language models we use a collection of 1.6 billion words, which we divide into 20 different sources. The two most important components are the broadcast news (**BN**) and broadcast conversation (**BC**) acoustic transcripts (7.5 million words each) corresponding to 1,800 h of speech transcribed by LDC for the GALE program. We use a vocabulary of 795,000 words, which is based on all available corpora, and is designed to completely cover the acoustic transcripts. To build the baseline language model, we train a 4-gram model with modified Kneser–Ney smoothing [13] for each source, and then linearly interpolate the 20 component models with the interpolation weights chosen to optimize perplexity on a held-out set. We combine all the 20 components into one language model using entropy pruning [48]. By varying the pruning thresholds we create (1) a 913 million n-gram LM (no pruning) to be used for lattice rescoring (**Base**) and (2) a 7 million n-gram LM to be used for the construction of static, finite-state decoding graphs.

In addition to the baseline language models described above, we investigated various other techniques which differ in either the features they employ or the modeling strategy they use. These are described below.

- **ModelM** A class-based exponential model [11]. Compared to the models used in the previous evaluation, we use a new enhanced word classing [12]. The bigram mutual information clustering method used to derive word classes in the original

Model M framework is less than optimal due to mismatches between the classing objective function and the actual LM, so the new method attempts to address this discrepancy. Key features of the new method include: (a) a class-based model that includes word n-gram features to better mimic the nature of the actual language modeling, (b) a novel technique for estimating the likelihood of unseen data for the clustering model, and (c) n-gram clustering compared to bigram clustering in the original method. We build Model M models with improved classing on 7 of the corpora with the highest interpolation weights in the baseline model.

- **WordNN** A 6-gram neural network language model using word features. Compared to the model used in the P4 evaluation [25], we train on more data (44 million words of data from **BN**, **BC** and **Archive**). We also enlarge the neural network architecture (increased the feature vector dimension from 30 to 120 and the number of hidden units from 100 to 800) and normalize the models. We create a new LM for lattice rescoring by interpolating this model with the 7 **ModelM** models and **Base**, with the interpolation weights optimized on the held-out set. In the previous evaluation we did not get an improvement by interpolating the WordNN model with model M models, but the changes made this year result in significant improvements.
- **SyntaxNN** A neural network language model using syntactic and morphological features [29]. The syntactic features include exposed head words and their non-terminal labels, both before and after the predicted word. For this neural network model we used the same training data and the same neural network architecture as the one described for **WordNN**. This language model is used for n-best rescoring.
- **DLM** A discriminative language model trained using the minimum Bayes risk (MBR) criterion [31]. Unlike the other LMs, a DLM is trained on patterns of confusion or errors made by the speech recognizer. Our potential features consist of unigram, bigram, and trigram morphs, and we used the perceptron algorithm to select a small set of useful features. With the selected features, we trained the DLM using an MBR-based algorithm, which minimizes the expected loss, calculated using the word error information and posterior probabilities of the N-best hypotheses of all the training sentences. To prepare data for DLM training, we used a Phase 3 unvowelized recognizer trained on 1,500 h of acoustic data to decode an unseen 300 h set. This unseen training set was provided in Phase 4, but adding this data to acoustic or language model training did not improve the system, so it is an ideal set for DLM training. During the evaluation, a single MBR DLM thus trained was used to rescore the N-best hypotheses from all the systems. Although there is a mismatch between training and test conditions, improvements were still observed. Details of these experiments as well as post-eval experiments are presented in [31].

Having many diverse language models, the challenge is to be able to combine them while achieving additive gains, and Sect. 13.4.4 describes our strategy.

13.4.3 System Combination

We employ three different techniques for system combination. The first technique is cross-adaptation ($\times$), where the fMLLR and MLLR transforms required by a speaker-adapted acoustic model are computed using transcripts from some other, different speaker-adapted acoustic model. The second technique is tree-array combination ($+$), a form of multi-stream acoustic modeling in which the acoustic scores are computed as a weighted sum of scores from two or more models that can have different decision trees [47]. The only requirement for the tree-array combination is that the individual models are built using the same pronunciation dictionary. The third technique is hypothesis combination using the `nbest-rover` [50] tool from the SRILM toolkit [49]. In all these combination strategies, the choice of systems to combine was based on performance on a variety of development sets.

13.4.4 System Architecture

IBM's 2011 GALE Arabic transcription system is a sequence of multiple passes of decoding, acoustic model adaptation, language model rescoring, and system combination steps. In this section, we show how all the models described in the previous sections are combined to generate the final transcripts. We report results on several data sets: DEV'07 (2.5 h); DEV'09 (2.8 h); EVAL'09, the unsequestered portion of the GALE Phase 4 evaluation set (4.2 h); and EVAL'11, the GALE Phase 5 evaluation set (3 h). EVAL'11 is unseen data on which no tuning was done. In our 2011 evaluation system we have the following steps.

1. Cluster the audio segments into hypothesized speakers.
2. Decode with the **SI** model.
3. Compute VTLN warp factors per speaker using transcripts from (2).
4. Decode using the **U** model cross-adapted on **SI**.
5. Decode using the **SGMM** model cross-adapted on (4).
6. Compute best frame rates per utterance using transcripts from (5).
7. Decode using the **SGMM** model cross-adapted on **U** and frame rates from (6).
8. a. Using the **U** model and transcripts from (5), compute fMLLR and MLLR transforms.
   b. Using the **BS** model and transcripts from (5), compute fMLLR and MLLR transforms.
   c. Using the **NNU** model and transcripts from (5), compute fMLLR and MLLR transforms.
   d. Decode and produce lattices using a tree-array combination of the **U** model with transforms from (8a), the **BS** model with transforms from (8b) and **NNU** with transforms from (8c).
9. a. Using the **M** model and transcripts from (5), compute fMLLR and MLLR transforms.

Table 13.14 Word error rates for the final three combined models before and after adding LM rescoring passes

Step	Decoding pass	DEV'09	EVAL'09	EVAL'11
(8d)	(U + BS + NNU) × SGMM × U	12.6 %	9.5 %	8.9 %
(9c)	(M + NNM) × SGMM × U	13.3 %	9.8 %	9.2 %
(10)	V × SGMM × U	13.5 %	9.8 %	9.5 %
(14a)	(8d) + simplex	11.4 %	8.4 %	7.8 %
(14b)	(9c) + simplex	11.9 %	8.6 %	8.1 %
(14c)	(10) + simplex	12.5 %	9.2 %	8.4 %
(15)	(14a) + (14b) + (14c)	11.1 %	8.1 %	7.4 %

b. Using the **NNM** model and transcripts from (5), compute fMLLR and MLLR transforms.
c. Decode and produce lattices using a tree-array combination of the **M** model with transforms from (9a) and the **NNM** model with transforms from (9b).

10. Decode and produce lattices using the **V** model, frame rates from (6) and fMLLR and MLLR transforms computed using transcripts from (5).
11. Using an interpolation of **Base**, 7 **ModelM** and one **WordNN** language models

    a. Rescore lattices from (8d), extract 50-best hypotheses
    b. Rescore lattices from (9c), extract 50-best hypotheses
    c. Rescore lattices from (10), extract 50-best hypotheses.

12. Parse the 50-best lists from (11) and score them with a **SyntaxNN** language model; produce new language model scores for each hypothesis.
13. Score the 50-best lists from (11) with a discriminative language model and produce new language model scores for each hypothesis.
14. Combine acoustic scores, language model scores from (11), syntax LM scores from (12) and discriminative LM scores from (13) using simplex

    a. Add the new scores to the hypotheses from (11a)
    b. Add the new scores to the hypotheses from (11b)
    c. Add the new scores to the hypotheses from (11c).

15. Combine the hypotheses from (14a), (14b), and (14c) using the `nbest-rover` tool from the SRILM toolkit [49]. This constitutes the final output (Table 13.14).

Table 13.15 shows the word error rates obtained after adding new language models either for lattice or n-best rescoring for the (8d) system. It can be seen that each additional rescoring pass improves the performance, and that the total improvement from language modeling rescoring is 1.1–1.2 % absolute on all the sets. Similar improvements have been obtained on the other two systems that are part of the final system combination ((9c) and (10)).

Table 13.15 Word error rates for different LM rescoring steps on the (U+BS+NNU)×SGMM× U.vfr (8d) set of lattices

Step	Language model	DEV'09	EVAL'09	EVAL'11
(8d)	Base	12.6 %	9.5 %	8.9 %
(11)	+ ModelM and WordNN	11.7 %	8.8 %	8.2 %
(12)	+ SyntaxNN	11.6 %	8.6 %	7.9 %
(13)	+ DLM	11.5 %	8.6 %	8.0 %
(14)	+ SyntaxNN and DLM	11.4 %	8.4 %	7.8 %

13.5 From MSA to Dialects

One of the key challenges in Arabic speech recognition research is how to handle the differences between Arabic dialects. Most recent work on Arabic ASR has addressed the problem of recognizing Modern Standard Arabic (MSA). Little work has focused on dialectal Arabic [26, 51]. Arabic dialects differ from MSA and each other in many dimensions of the linguistic spectrum, morphologically, lexically, syntactically, and phonologically. What makes Arabic dialect challenging in particular is the lack of a well-defined spelling system, resources (i.e., acoustic and LM training data) as well as tools (such as morphological analyzers and disambiguation tools).

In this section, we report a series of experiments about how we can progress from Modern Standard Arabic (MSA) to Levantine ASR, in the context of the GALE DARPA program. While our GALE models achieved very low error rates, we still see error rates twice as high when decoding dialectal data. We make use of a state-of-the-art Arabic dialect recognition system to automatically identify Levantine and MSA subsets in mixed speech of a variety of dialects including MSA. Training separate models on these subsets, we show a significant reduction in word error rate over using the entire data set to train one system for both dialects. During decoding, we use a tree array structure to mix Levantine and MSA models automatically using the posterior probabilities of the dialect classifier as soft weights. This technique allows us to mix these models without sacrificing performance for either variety. Furthermore, using the initial acoustic-based dialect recognition system's output, we show that we can bootstrap a text-based dialect classifier and use it to identify relevant text data for building Levantine language models.

13.5.1 *Dialect Identification*

As mentioned above, we are interested in building Levantine-specific models using the available GALE data. Recall that this data contains a mix of dialects in addition to MSA and that this data has no specific dialect annotations. To build a Levantine-specific ASR system, we need dialect annotations for each utterance since Arabic speakers, in broadcast conversations (BC), tend to code mix/switch between MSA and their native dialects across utterances and even within the same utterance.[3] In this work, we build a dialect recognition system to identify dialects at the utterance level.

Biadsy et al. [4, 6] have previously shown that a dialect recognition approach that relies on the hypothesis that certain phones are realized differently across dialects achieves state-of-the-art performance for multiple dialect and accent tasks (including Arabic). We make use of this system (described next) to annotate some of our Arabic GALE data.

Phone Recognizer and Front-End

The dialect recognition approach makes use of phone hypotheses. Therefore, we first build a triphone context-dependent phone recognizer. The phone recognizer is trained on MSA using 50 h of GALE speech data of broadcast news and conversations with a total of 20,000 Gaussians. We use one acoustic model for silence, one for non-vocal noise and another to model vocal noise. We utilize a unigram phone model trained on MSA to avoid bias for any particular dialect.[4] We also use FMLLR adaptation using the top CD-phone sequence hypothesis. Our phone inventory includes 34 phones, 6 vowels and 28 consonants.

Phone GMM-UBM and Phonetic Representation

The first step in the dialect recognition approach is to build a 'universal' acoustic model for each context-independent phone type. In particular, we first extract acoustic features (40d feature vectors after CMVN and FMLLR) aligned to each phone instance in the training data (a mix of dialects). Afterwards, using the frames aligned to the same phone type (in all training utterances), we train a Gaussian Mixture Model (GMM), with 100 Gaussian components with diagonal covariance matrices, for this phone type, employing the EM algorithm. Therefore, we build 34 GMMs. Each phone GMM can be viewed as a GMM-Universal Background Model (GMM-UBM) for that phone type, since it models the general realization of that phone across dialect classes [38]. We call these GMMs *phone GMM-UBMs*.

Each phone type in a given utterance (U) is represented with a single MAP (Maximum A-Posteriori) adapted GMM. Specifically, we first obtain the acoustic frames aligned to every phone instance of the same phone type in U. Then these frames are used to MAP adapt the means of the corresponding phone GMM-UBM

[3]In this work, we do not attempt to identify code switching points; we simply assume that an utterance is spoken either in MSA or in purely a regional dialect.

[4]We use true phonetic labels here by generating pronunciation dictionaries using MADA, following [4].

using a relevance factor of $r = 0.1$. The resulting GMM of phone type ϕ is called the *adapted phone-GMM* (f_ϕ). The intuition here is that f_ϕ 'summarizes' the variable number of acoustic frames of all the phone instances of a phone-type ϕ in a new distribution specific to ϕ in U [6].

A Phone-Type-Based SVM Kernel

Now, each utterance U can be represented as a set S_U of adapted phone-GMMs, each of which corresponds to one phone type. Therefore, the size of S_U is at most the size of the phone inventory ($|\Phi|$). Let $S_{U_a} = \{f_\phi\}_{\phi \in \Phi}$ and $S_{U_b} = \{g_\phi\}_{\phi \in \Phi}$ be the adapted phone-GMM sets of utterances U_a and U_b, respectively. Using the kernel function in Eq. (13.16), designed by Biadsy et al. [6] which employed the upper bound of KL-divergence-based kernel (13.17), proposed by Campbell et al. [9], we train a binary SVM classifier for each pair of dialects. This kernel function compares the 'general' realization of the same phone types across a pair of utterances.

$$K(S_{U_a}, S_{U_b}) = \sum_{\phi \in \Phi} K_\phi(f'_\phi, g'_\phi) \tag{13.16}$$

where f'_ϕ is the same as f_ϕ but we subtract from its Gaussian mean vectors the corresponding Gaussian mean vectors of the phone GMM-UBM (of phone type ϕ). g'_ϕ is obtained similarly from g_ϕ. The subtraction forces zero contributions from Gaussians that are not affected by the MAP adaptation. And,

$$K_\phi(f_\phi, g_\phi) = \sum_i (\sqrt{\omega_{\phi,i}}\, \Sigma_{\phi,i}^{-\frac{1}{2}} \mu_i^f)^T (\sqrt{\omega_{\phi,i}}\, \Sigma_{\phi,i}^{-\frac{1}{2}} \mu_i^g) \tag{13.17}$$

where, $\omega_{\phi,i}$ and $\Sigma_{\phi,i}$ respectively are the weight and diagonal covariance matrix of Gaussian i of the phone GMM-UBM of phone-type ϕ; μ_i^f and μ_i^g are the mean vectors of Gaussian i of the *adapted* phone-GMMs f_ϕ and g_ϕ, respectively.

It is interesting to note that, for (13.16), when K_ϕ is a linear kernel, such as the one in (13.17), each utterance S_{U_x} can be represented as a single vector. This vector, say W_x, is formed by stacking the mean vectors of the adapted phone-GMM (after scaling by $\sqrt{\omega_\phi}\, \Sigma_\phi^{-\frac{1}{2}}$ and subtracting the corresponding μ_ϕ) in some (arbitrary) fixed order, and zero mean vectors for phone types not in U_x. This representation allows the kernel in (13.16) to be written as in (13.18). This vector representation can be viewed as the 'phonetic fingerprint' of the speaker. It should be noted that, in this vector, the phones constrain which Gaussians can be affected by the MAP adaptation (allowing comparison under linguistic constraints realized by the phone recognizer), whereas in the GMM-supervector approach [8], in theory, any Gaussian can be affected by any frame of any phone.

$$K(S_{U_a}, S_{U_b}) = W_a^T W_b \tag{13.18}$$

Table 13.16 F-Measure for each dialect class using the four-way classifier with 30 s cuts

Dialect	F-measure
Levantine	90.7 %
Iraqi	86.7 %
Gulf	87.1 %
Egyptian	98.6 %

13.5.2 ASR and Dialect ID Data Selection

As noted above, the GALE data is not annotated based on dialects. Moreover, to the best of our knowledge, there is no Arabic dialect corpus of similar domain and/or acoustic condition as broadcast conversations. Fortunately, there are telephone conversation corpora available from the LDC for four Arabic dialects (Egyptian, Levantine, Gulf, and Iraqi). To address the acoustic recording and domain issues we build two systems.

In our first system, we train our dialect recognition on dialect data taken from spontaneous telephone conversations from the following Appen corpora: Iraqi Arabic (478 speakers), Gulf (976), and Levantine (985). For Egyptian, we use the 280 speakers in CallHome Egyptian and its supplement. Using the kernel-based approach describe above, we train a binary SVM classifier for each pair of dialects on 30 s cuts of 80 % of the speakers (of each corpus). Each cut consists of consecutive speech segments totaling 30 s in length (after removing silence). Multiple cuts are extracted from each speaker.[5] As a result, we obtain six binary classifiers for the four broad Arabic dialects.

To label utterances with dialect ID tags, we need a single four-way classifier to classify the dialect of the speaker to one of the four dialects. To build such a classifier, we first run the six SVM binary classifiers on the remaining 20 % held-out speakers. Every SVM binary classifier provides a posterior probability $P(C_1|x)$ for each test sample x of belonging to class C_1. We use these posteriors as features to train a four-way logistic regression on the 20 % set. The 10-fold cross validation of this classifier is 93.3 %; the F-measure of each dialect class is shown in Table 13.16.

We run this system to annotate a portion of our GALE BC data (after down-sampling to 8 Khz). The dialect recognition system classified 54 h of Levantine speech with a relatively high confidence. Since the dialect ID system is trained on telephone conversations as opposed to broadcast conversations, we asked the LDC to validate/filter the output of the system. We find that about 36 h out of 54 h are tagged as "mostly Levantine", a 10 h set contains code switching between MSA and Levantine at the utterance level, and an 8 h set contains either other dialects or MSA. Recall that our system is not trained to identify MSA.

We extract a 4 h test set (**LEV_4h**) to be used for reporting results in all the Levantine ASR experiments. From the remaining 32 h we extract all the utterances

[5]The equal error rate reported by Biadsy et al. [6] of this dialect recognition system on the remaining 20 % held-out speakers is 4 %.

Table 13.17 MADA AM used for dialect ID, WER test

System	WER on DEV-07
50k Gaussians, 1k states, ML	16.8 %
200k Gaussians, 5k states, ML	15.4 %
200k Gaussians, 5k states, fBMMI + BMMI	12.5 %

longer than 20 s, this yields approximately 10 h of data (**LEV_10**). Part of the transcripts released by LDC for the GALE program have "non-MSA" annotations. This allows us to select a 40 h MSA corpus by choosing speakers whose utterances have no such markings. From this set we select 4 h for our MSA ASR experiments (**MSA_4h**). From the remining data, we further select a 10 h set with utterances longer than 20 s (**MSA_10**).

13.5.3 Dialect Identification on GALE Data

Given that now we have gold standard BC MSA and Levantine data (MSA_10 and LEV_10), we can train another dialect recognition system to distinguish MSA vs. Levantine for BC acoustic conditions. We divide LEV_10 into 9 h for training and 1 h for testing our dialect recognition system. Similarly MSA_10 is divided into 9 h for training and 1 h for testing. Note that this amount of acoustic data is typically not sufficient to train dialect identification systems; however, we are interested in making use of the rest of the data for other experiments.

As described in Sect. 13.5.1, for the dialect identification system we need a phone decoder; therefore we carry out a number of experiments to find the best strategy for building it. We train three MADA Vowelized (i.e., a true phonetic-based system) triphone acoustic models in which we vary the number of Gaussians and the number of states, using either ML or discriminative training. First, we test these models for word recognition with our unpruned 4-gram LM. Table 13.17 shows the word error rates on the DEV-07 set.

In the next test, we use the triphone models to decode phone sequences with different phone language models. For each phone decoder we build a dialect classification system using the SVM-Kernel approach described in Sect. 13.5.1. We train the models on 9 h of Levantine data and 9 h of MSA data, and evaluate the results on a test set which contains 1 h of Levantine and 1 h of MSA data. Table 13.18 shows the dialect classification rates for the different acoustic model and phone language model combinations. Based on these results we decided to use the smallest, simplest model (50k Gaussians ML model with unigram phone language model) for the subsequent experiments.

Table 13.18 Dialect classification performance

System/features	Classification accuracy
50k ML 1-gram phone LM	85.1 %
50k ML 3-gram phone LM	84.5 %
200k ML, 3-gram phone LM	84.9 %
200k fBMMI + BMMI, 3-gram	83.0 %

Table 13.19 300 h AM tested on DEV-07

System	Unvowelized	BW vowelized	MADA vowelized
ML	16.6 %	14.2 %	13.9 %
fBMMI + BMMI	12.7 %	11.8 %	11.7 %

13.5.4 Acoustic Modeling Experiments

Comparing Vowelizations

We select a 300-h subset from our entire GALE training set and train speaker adaptive acoustic models for all three lexical setups. The decoding setup includes VTLN, FMLLR, and MLLR and we use an unpruned 4-gram LM with a 795k vocabulary. First, we test the models on one of our standard GALE development sets, DEV-07, shown in Table 13.19. Pronunciation probabilities are used for both, Buckwalter and MADA systems. Buckwalter and MADA vowelizations perform similarly, while the unvowelized models are 2.7 % worse at the ML level. However, we want to note that the difference is only 1 % after discriminative training. This indicates that discriminative training of context-dependent (CD) GMM models is able to compensate for the lack of (knowledge based) pronunciation modeling to a large degree.

In the next comparison, we test the models on a newly defined MSA test set. The reason behind this set is that we want to use the same methodology for defining/selecting a test set for both Levantine and MSA. We would like to analyze the difficulty of Levantine when compared to MSA under exactly same conditions. We are basically reducing effects related to how and from where the test sets are chosen. DEV-07, for example, is a test set defined by LDC which consists of mostly very clean broadcast news data. This is very likely the reason behind our very low error rates. The MSA_4h test set is selected randomly from broadcast conversations of our training set and labeled as MSA by our dialect classifier. The reason to select the data from broadcast conversations is to match the conditions of the Levantine test set. All of the Levantine data comes from BC as well. The error rates on this MSA test set (Table 13.20) is almost twice as high as the error rates on DEV-07 (Table 13.19), although both are non-dialectal (MSA) test data. We also see that all three models perform at a similar level (21.2–21.8 %) after discriminative training.

We now compare the models on Levantine data (LEV_4). Recall that this Levantine test set is part of the GALE corpus identified automatically by our dialect classifier and manually verified by LDC (see Section above). The same

Table 13.20 300 h AM tested on MSA_4h

System	Unvowelized	BW vowelized	MADA vowelized
ML	28.6 %	27.0 %	25.7 %
fBMMI + BMMI	21.8 %	21.7 %	21.2 %

Table 13.21 300 h AM tested on LEV_4h

System	Unvowelized	BW vowelized	MADA vowelized
ML	48.2 %	50.3 %	48.1 %
fBMMI + BMMI	39.7 %	42.1 %	40.8 %

methodology for selecting the test data is used for MSA_4h and LEV_4h. Both MSA_4h and LEV_4h test sets are excluded from the training of the acoustic and language models. Looking at Tables 13.20 and 13.21, we observe two main points:

1. The error rate for Levantine is almost twice as high as for MSA (39.7 vs. 21.8 %). We compare here the Levantine error rate to MSA_4h and not to DEV-07. This allows us to attribute the increase in error rate to dialect and not to other effects (how the test set was chosen and how carefully the transcripts were done).
2. Another interesting observation is that the unvowelized models perform best on Levantine (39.4 vs. 40.8 and 42.1 %). We speculate that this is due to the fact that the Buckwalter analyzer, MADA, and the pronunciation rules are designed for MSA – which do not work properly for Levantine words. A dialect-specific morphological analyzer would very likely improve results, but it is unclear that it would significantly reduce the error rate on Levantine given that the unvowelized perform comparably well on MSA data (Table 13.20).

Selecting Dialect Data from the 300-h Training Subset

We now run the dialect recognition system on our 300-h subset of the GALE training corpus. Out of this training set, we obtain about 37 h labeled as Levantine. This is not sufficient to train a set of acoustic models. One option is to use a deep MLLR regression tree or MAP training. In our experience MLLR works well for limited domain adaptation data, but will not be able to fully utilize a large amount of domain adaptation data. While MAP works better with more adaptation data, it is difficult to use it in combination with feature space discriminative training.

Instead, we use a form of training with weighted statistics. The advantage is that all components of the model (including decision trees) are trained at all training stages (ML, DT) with the new domain data. In our case we have additional information in the form of a dialect posterior probability for each utterance from the dialect classifier. We use this posterior to weight the statistics of each utterance during ML and discriminative training. The modified statistics are computed as shown in formula (13.19).

Table 13.22 Comparing weighting schemes of training statistics on LEV_4h, 300 h setup, unvowelized ML models

Training data	WER
Unweighted (300 h)	48.2 %
Hard-weighted (37 h)	48.3 %
Soft-weighted (300 h)	45.3 %

Table 13.23 300 h AM tested on LEV_4h

System	Unvowelized	BW vowelized	MADA vowelized
ML	45.3 %	47.3 %	45.5 %
fBMMI + BMMI	38.4 %	41.4 %	39.2 %

$$E(x) = \sum_i P(dialect|x_i) * x_i$$

$$E(x^2) = \sum_i P(dialect|x_i) * x_i^2 \qquad (13.19)$$

Table 13.22 shows a comparison of different weighting schemes. In the first row, we simply train on all 300 h regardless whether they are Levantine or MSA. This model gives us an error rate of 48.2 %. In the second row, we train only on the selected Levantine subset of 37 h. The error rate is slightly higher, 48.3 %, due to the lack of training data. In the third row, we train on the same 300 h, but weight the statistics of each utterance individually by the posterior score of the dialect classifier. This provides us with a smoothing of the models, avoids overtraining and we get a 2.9 % error reduction.

We apply now the soft-weighting scheme to all vowelization setups and compare the models both after ML and fBMMI + BMMI training in Table 13.23. The improvement from focusing on Levantine training data can be seen by comparing Table 13.21 with Table 13.23. For example, for the unvowelized models, we obtain 2.9 % absolute error reduction at the ML level, and 1.3 % after discriminative training. Note that we do not add training data, rather we find relevant subsets that match our target dialect.

Tree Array Combination

When we focus the training on Levantine, we can expect the model to perform worse on MSA data. In fact, the error rate increases from 12.7 to 15.1 % on DEV-07 when we use the Levantine models (Table 13.24). Our toolkit allows us to combine models with different decision trees into one single decoding graph [46]. This enables us to combine different acoustic models in one decoding pass on the fly, without making a hard model selection. The combined acoustic score is the weighted sum of the log likelihoods of the combined models. In our case, we combine the MSA and LEV unvowelized models. The results are in Table 13.24. The first two rows represent

Table 13.24 Tree array combination of general models with Levantine models in on decoding pass, 300-h unvowelized fBMMI + BMMI setup

Weight for MSA model	Weight for LEV models	DEV-07	LEV_4h
1.0	0.0	12.7 %	39.7 %
0.0	1.0	15.1 %	38.4 %
0.5	0.5	13.3 %	38.2 %
Dialect classifier soft weight		12.9 %	38.4 %

Table 13.25 Comparing weighting schemes of training statistics on LEV_4h, 1,800-h setup, unvowelized ML models

Training data	WER
Unweighted (1,800 h)	47.0 %
Hard-weighted (237 h)	42.3 %
Soft-weighted (1,800 h)	43.5 %

the extreme cases where either the MSA or LEV model is used exclusively. In the third row, we weight both models equally and constant for all utterances. The error rate on DEV-07 is 13.3 %, 0.6 % higher than when just using the MSA model, but much better than when using the LEV models only (15.1 %). On the other hand, we get a small improvement on the Levantine test set (38.4 % goes to 38.2 %). This is a system combination effect. We used tree arrays in the past as an alternative to ROVER or cross-adaptation, for example in our latest GALE evaluation. In the last row in Table 13.24 we use the posterior of the dialect classifier as a soft weight for model combination on a per utterance basis. This automatic strategy gives us an error rate that is close to the optimal performance of a model selected manually.

Selecting Dialect Data from the 1,800-h Training Set

The full GALE training corpus consists of about 18,00 h. Similar to the previous experiments, but now focusing exclusively on the unvowelized models, we generate dialect labels for the entire training corpus. The dialect recognition system identified about 237 h as Levantine in the GALE corpus (or 13 %). In Table 13.25, we compare different weighting schemes for the Levantine data. In contrast to the 300 h setup (Table 13.22), the best error rate is achieved now by training exclusively on the 237 h of Levantine data and not by using the dialect scores to weight the statistics. The reason is simply that the amount of Levantine training data is now large enough to train acoustic models and we do not need to add data as was the case for the previous experiments when we had only 37 h of Levantine data.

After discriminative training (fBMMI + bMMI) of the 237 h unvowelized Levantine models, the error rate goes down to 36.3 %. In other words, we can lower the error rate by almost 10 % relative by focusing on relevant subsets of the training data and the dialect classifier together with the tree array decoding technique which allows us to use both Levantine and MSA models in one decoding pass, so the engine can handle both dialectal and non-dialectal utterances at the same time.

Table 13.26 Text only dialect classification using Levantine and MSA language models

Test data	Dialect classification
MSA_4h	86.0 %
Lev_4h	87.2 %

Table 13.27 LM rescoring with Levantine LM

Training data	WER
913m 4-gram baseline LM	36.3 %
+ 3-gram Levantine LM from 238 h set	35.4 %
+ 4-gram Levantine weighted LM (all text sources)	35.1 %

13.5.5 Dialect ID Based on Text Only

The experiments described in Sect. 13.5.4 demonstrate that the acoustic training data contains relevant dialect subsets which when detected can improve the acoustic models. In this section, we report on a similar strategy for language modeling, but now we built a dialect classifier based on text only – no audio data is used. First, we build a Kneser–Ney smoothed 3-gram Levantine LM on the 2 million words corresponding to the transcripts of the 237 h of Levantine acoustic training data (identified automatically). Similarly, we build an MSA language model from all the utterances which are classified as MSA with more than 95 % probability by the dialect annotator. We build a text dialect classifier which simply checks the log-likelihood ratio of the two LMs on a given utterance. Table 13.26 shows that we can predict the dialect reliably even when only text data is available.

Levantine LM

Our GALE language models are trained on a collection of 1.6 billion words, which we divide into 20 parts based on the source. We train a 4-gram model with modified Kneser–Ney smoothing [13] for each source, and then linearly interpolate the 20 component models with the interpolation weights chosen to optimize perplexity on a held-out set. In order to build a Levantine language model, we run the text dialect annotator described above on each of the 20 text sources and build 4-gram language models on the 20 dialectal subparts. The new 20 dialect language models are interpolated with the 20 original ones. We optimize the interpolation weights of the 40 language models on a Levantine held-out set. Table 13.27 shows the improvements obtained by adding dialect data to the original language model. Note that the improvement from adding dialect language models is less than the one obtained from dialect acoustic models. One reason for this is the fact that the initial dialect data is selected from the BC part of the training data, and the BC language model has a high weight in the baseline interpolated LM.

Finding Levantine Words

We can identify dialectal words if we compute how many times the word occurs in the Levantine corpus vs. the MSA one. After sorting the count ratios, we find the following words ranked at the top of the list: Em, hyk, bdw, bdk, ylly, blbnAn, which are in fact Levantine words. Note that identifying dialectal words can be useful for building better pronunciation dictionaries for dialects as well as for machine translation.

13.6 Resources

As explained in Sect. 13.1.1 a speech recognizer consists of models representing the underlying acoustic and language characteristics. In order to build a speech recognizer, language-specific data for training these models is required. The Linguistic Data Consortium (www.ldc.upenn.edu) is a good starting point to look for publically available data resources. In this section we describe the available data for Arabic that was mostly collected as part of DARPA programs.

13.6.1 Acoustic Training Data

We used the following corpora for acoustic model training in various experiments presented here:

- 85 h of FBIS and TDT-4 audio with transcripts provided by BBN,
- 51 h of transcribed GALE data provided by the LDC for the GALE Phase 1 (P1) evaluation,
- 700 h of transcribed GALE data provided by LDC for the Phase 2 (P2) and Phase 2.5 (P2.5) evaluations,
- 5.6 h of BN data with manually vowelized transcripts provided by BBN (BNAT-05),
- 5.1 h of BN data with manually vowelized transcripts provided by BBN (BNAD-05),
- 500 h of transcribed Iraqi data (TRANSTAC),
- 1,800 h of untranscribed audio from the EARS BN-03 corpus, and
- 10,000 h of untranscribed audio collected at IBM Research (TALES).

BNAT-05 and BNAD-05 were used only for a comparison of flat-start training to manual data for the initialization of vowelized models. The TRANSTAC data was used only for experiments on dialect modeling. The TALES data was used only for experiments on large-scale unsupervised training, including tests of dialect models. The evaluation system described in Sect. 13.4 was trained only on data available to all GALE program participants: FBIS, TDT-4, GALE P1 and P2, and EARS BN-03.

Table 13.28 Source variability in Arabic broadcast training data

Source	Hours
Al Arabiya	2,455
Al Jazeera Morning News	306
Al Jazeera Midday News	326
Al Jazeera News Bulletin	975
Dubai News	27
ESC News	81
LBC (Lebanese) Flash News	88
LBC International News	161
Voice of America	184
Al Alam	47
NBN News	131

Arabic broadcast news and conversations are highly variable, coming from many different broadcasters and containing a mixture of Modern Standard Arabic (MSA) and dialectical Arabic. To illustrate this variability, Table 13.28 provides a breakdown of a sample comprising 130 h of GALE transcribed data, 750 h of untranscribed EARS BN-03 data, and 5,600 h of untranscribed TALES data by source. Note that only the predominant sources in the sample are listed; there is also a long tail containing many additional sources which are represented by smaller amounts of audio.

13.6.2 Training Data for Language Modeling

We used the following resources for language modeling:

- Transcripts of the audio data released by LDC (7M words),
- The Arabic Gigaword corpus (500M words),
- News group and web log data collected by LDC (22M words),
- Web transcripts for broadcast conversations collected by CMU/ISL (100M words), and
- web text data collected by Cambridge University and CMU (200M words).

Automatic transcripts from the unsupervised training corpus were not used for language modeling.

13.6.3 Vowelization Resources

Besides the training data, there are other resources that are helpful to build systems for Arabic, but not necessarly essential. Many research groups use these tools for their Arabic LVCSR systems to generate vowelized pronunciation lexica.

In Sect. 13.2.2 we demonstrate how these tools can be used to generate vowelized forms for unvowelized corpora and improve the speech recognition performance.

1. Buckwalter's morphological analyzer
   The morphological analyzer [7] can be used to decompose (unvowelized) words into morphemes and provide a list of vowelizations. It is also available from LDC.
2. MADA (Morphological Analysis and Disambiguation for Arabic)
   This tool [22] can be used to rank and refine the vowelized variants by performing a sentence-wide analysis. It uses the results from Buckwalter's morphological analyzer to rerank them using an SVM classifier that uses morphological and n-gram features.
3. Arabic Treebank
   The Arabic Treebank (www.ircs.upenn.edu/arabic) consists of part-of-speech, morphological and syntax annotations for a corpus collected as part of the DARPA TIDES program. The corpus can be used to find vowelized forms for pronucuation lexica.

13.7 Comparing Arabic and Hebrew ASR

The goal of this section is to describe the challenges involved in building an ASR system for modern Hebrew. We hypothesize that most of the techniques used for building an Arabic ASR will carry over to Hebrew as well.

Hebrew is a morphologically rich language, therefore, as in Arabic, the vocabulary size can be very large, if explicit morphological modeling is not employed, affecting the lexicon size as well as the language model. All of this may highly likely lead to higher language model perplexity, larger decoder search space, and large memory requirements. We have seen in this chapter how to handle these issues for Arabic, hence the same techniques are likely to carry over. The diacritizaion phenomenon, or so-called Niqqud in Hebrew, is almost identical to that of Arabic. Short vowels and consonant alternation markers are written as diacritic markers above, below or inside letters. Similar to Arabic, most modern Hebrew texts are written with almost no Niqqud whatsoever. The same word with different Niqqud typically changes meaning, POS, and/or pronunciations, as in Arabic.

As in Arabic, there is almost a one-to-one mapping between fully diacritized Hebrew words to pronunciations using simple pronunciation rules. Therefore, automatic morphology analysis and disambiguation techniques may be useful for Hebrew to build a "vowelized" system as we have seen for Arabic. We also hypothesize that a completely "unvowelized" system, as in Arabic, will also perform relatively well for Hebrew. In other words, ignoring the Niqqud completely may provide us with a relatively accurate system, particularly when discriminative training is used. Unlike Arabic, in modern Hebrew, there is no notion of regional dialects. This suggests that building a Hebrew ASR system is far easier than building a unified Arabic ASR system. This is due to the facts that Hebrew has a single well-defined spelling system, a single vocabulary set, a single morphological and syntactic system and a single phonetic inventory.

13.8 Summary

We described in this chapter LVCSR techniques, both language-independent and language-specific, to obtain high performance LVCSR systems. Our main findings are based on our experience from the 5-year DARPA GALE program, aimed at improving Arabic speech recognition and translations. Our main findings are:

1. Current LVCSR technology works fairly robustly on a set of different languages. Ninety percent of the improvements in our system come from basic technology that works across languages. It is important to get the basic techniques for acoustic and language modeling right before addressing language-specific issues.
2. The lack of diacritics in written texts is one of the more important language-specific issues. Short vowels are not written but spoken, creating a mismatch between transcripts and audio files. The first solution is to use semi-automatic vowelization procedures such as Buckwalter's morphological analyzer. These tools can be used to add vowelized pronunciation variants to the lexicon. Flat-start training over pronunciation graphs can then initialize acoustic models. Repeated system building will lead to very good vowelized acoustic models.
3. Discriminative training is able to compensate for modeling errors. Instead of trying to model short vowels, we explore the idea of letting our acoustic models learn to deal with modeling errors in the pronciation lexicon. Long-span phonetic context allows decision trees to create states corresponding to effects related to missing short vowels. Discriminative training (feature and model space boosted MMI or MPE) trains acoustic models in a way that corrects modeling errors. Experimental results show that unvowelized models are very competitive once speaker adaptation and discriminative training is added.
4. The rich morphology of the Arabic language is another issue that is important for LVCSR. The rich morphology creates two problems. The first problem is that a lot of words are not covered by regular sized vocabularies. The second problem is data sparsity for language modeling. One solution for these problems is to build a morpheme-based vocabulary and language model. This approach yields very low OOV rates while maintaining regular sized vocabularies. The downside is a postprocessing step that converts the recognition output back to words. This postprocessing step will create additional errors since the mapping is usually ambiguous.

   A more elegant approach, in our opinion, is to rely on careful engineering. A well-written LVCSR decoder can easily work with very large vocabularies. Then, we can use word-based vocabularies and simply increase the size of the vocabulary to millions of words. No postprocessing of the recognition output is required and the OOV rates on standard tests are below 0.5 %. Class-based exponential language models such as Model-M have proved to be very capable.

5. Increased data sets make language-specific techniques less important. During the course of the GALE program the amount of training data increased from 100 h to more than 1,300 h. We observed that improvements from vowelization were reduced significantly when more training data became available.
6. Languages such as Arabic cover a wide variety of dialects. We presented several methods for helping LVCSR systems cope with dialects. One approach is to make acoustic models dialect specific. Adding dialect questions to the decision tree is a data-driven way to generate HMM states specific to certain dialects. Another approach is to bootstrap a dialect-specific system by leveraging MSA models. The dialect recognition system allows us to focus on relevant training subsets. While specialized models for Levantine perform poorly on MSA, the tree array decoding procedure allows us to mix both models without sacrificing performance. Also, we showed that we can build a text-only dialect classifier that performs as well as a dialect classifier requiring audio data. The text-only dialect classifier enables us to find relevant LM text data. Another application is pronunciation modeling where the text-based dialect classifier can provide us with candidate words that occur only in Levantine.
7. Diversity of acoustic models is good for system combination. Instead of relying on either vowelized or unvowelized models, we use both models. The combination can be done as cross-adapation, confusion network combination or other methods. System combination makes LVCSR technology more robust on a variety of test sets and improves the error rate significantly.

Acknowledgements We would like to acknowledge the support of DARPA under Grant HR0011-06-2-0001 for funding part of this work. The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

References

1. Kneser R, Ney H (1995) Improved backing-off for m-gram language modeling. In: International conference on acoustics, speech, and signal processing. ICASSP-95, Detroit, vol I, pp 181–184
2. Afify M, Nguyen L, Xiang B, Abdou S, Makhoul J (2005) Recent progress in Arabic broadcast news transcription at BBN. In: Proceedings of the Interspeech, Lisbon, pp 1637–1640
3. Bengio Y, Ducharme R, Vincent P, Jauvin C (2003) A neural probabilistic language model. J Mach Learn Res 3:1137–1155
4. Biadsy F (2011) Automatic dialect and accent recognition and its application to speech recognition. PhD. thesis, Columbia University
5. Biadsy F, Habash N, Hirschberg J (2009) Improving the Arabic pronunciation dictionary for phone and word recognition with linguistically-based pronunciation rules. In: Proceedings of NAACL/HLT 2009, Colorado

6. Biadsy F, Hirschberg J, Ellis D (2011) Dialect and accent recognition using phonetic-segmentation supervectors. In: Interspeech, Florence
7. Buckwalter T (2004) LDC2004L02: Buckwalter Arabic morphological analyzer version 2.0. Linguistic Data Consortium, Philadelphia
8. Campbell W, Sturim D, Reynolds D (2006) Support vector machines using GMM supervectors for speaker verification. IEEE Signal Process Lett 13(5):308–311
9. Campbell W, Sturim D, Reynolds D, Solomonoff A (2006) SVM based speaker verification using a GMM supervector kernel and NAP variability compensation. In: Proceedings of the ICASSP, France
10. Chelba C, Jelinek F (1998) Exploiting syntactic structure for language modeling. In: Proceedings of the 36th annual meeting of the Association for Computational Linguistics and 17th international conference on computational linguistics, Montreal, pp 225–231
11. Chen SF (2009) Shrinking exponential language models. In: Proceedings of the NAACL-HLT, Boulder
12. Chen S, Chu S (2010) Enhanced word classing for model M. In: Proceedings of the Interspeech, Makuhari, pp 1037–1040
13. Chen SF, Goodman JT (1998) An empirical study of smoothing techniques for language modeling. Technical report TR-10-98, Harvard University
14. Chu S, Povey D, Kuo HK, Mangu L, Zhang S, Shi Q, Qin Y (2010) The 2009 IBM GALE Mandarin broadcast transcription system. In: Proceedings of the ICASSP, Dallas, pp 4374–4377
15. Collins M, Roark B, Saraclar M (2005) Discriminative syntactic language modeling for speech recognition. In: Proceedings of the 43rd annual meeting of the Association for Computational Linguistics, Ann Arbor, pp 507–514
16. Davis SB, Mermelstein P (1980) Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. IEEE Trans Acoust Speech Signal Process 28:357–366
17. Emami A, Jelinek F (2005) A neural syntactic language model. Mach Learn 60:195–227
18. Emami A, Mangu L (2007) Empirical study of neural network language models for Arabic speech recognition. In: Proceedings of the ASRU 2007, Kyoto, pp 147–152
19. Fügen C, Rogina I (2000) Integrating dynamic speech modalities into context decision trees. In: Proceedings of the ICASSP, Istanbul
20. Gales MJF (1999) Semi-tied covariance matrices for hidden Markov models. IEEE Trans Speech Audio Process 7(3):272–281
21. Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the AISTATS, Sardinia, pp 249–256
22. Habash N, Rambow O (2005) Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In: Proceedings of the 43rd annual meeting of the Association for Computational Linguistics (ACL'05), Ann Arbor. Association for Computational Linguistics, pp 573–580. http://www.aclweb.org/anthology/P/P05/P05-1071
23. Habash N, Rambow O (2007) Arabic diacritization through full morphological tagging. In: NAACL07, Rochester
24. Kingsbury B (2009) Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling. In: Proceedings of the ICASSP, Taipei, pp 3761–3764
25. Kingsbury B, Soltau H, Saon G, Chu S, Kuo HK, Mangu L, Ravuri S, Morgan N, Janin A (2011) The IBM 2009 GALE Arabic speech transcription system. In: Proceedings of the ICASSP, Prague, pp 4378–4381
26. Kirchhoff K, Vergyri D (2004) Cross-dialectal acoustic data sharing for Arabic speech recognition. In: ICASSP, Montreal
27. Kirchhoff K, Bilmes J, Das S, Duta N, Egan M, Ji G, He F, Henderson J, Liu D, Noamany M, Schone P, Schwartz R, Vergyri D (2003) Novel approaches to Arabic speech recognition: report from the 2002 Johns-Hopkins summer workshop. In: Proceedings of the ICASSP, Hong Kong, pp 344–347

28. Kirchhoff K, Vergyri D, Bilmes J, Duh K, Stolcke A (2006) Morphology-based language modeling for conversational Arabic speech recognition. Comput Speech Lang 20(4):589–608
29. Kuo HKJ, Mangu L, Emami A, Zitouni I, Lee YS (2009) Syntactic features for Arabic speech recognition. In: Proceedings of the ASRU 2009, Merano
30. Kuo HKJ, Mangu L, Emami A, Zitouni I (2010) Morphological and syntactic features for Arabic speech recognition. In: Proceedings of the ICASSP 2010, Dallas
31. Kuo H, Mangu L, Arisoy E, Saon G (2011) Minimum Bayes risk discriminative language models for Arabic speech recognition. In: Proceedings of the of IEEE ASRU, Waikoloa
32. Maamouri M, Bies A, Buckwalter T, Mekki W (2004) The Penn Arabic Treebank: building a large-scale annotated Arabic corpus. In: Proceedings of NEMLAR conference on Arabic language resources and tools, Cairo, pp 1137–1155
33. Messaoudi A, Gauvain JL, Lamel L (2006) Arabic broadcast news transcription using a one million word vocalized vocabulary. In: Proceedings of the ICASSP, Toulouse, pp 1093–1096
34. Povey D, Woodland PC (2002) Minimum phone error and I-smoothing for improved discriminative training. In: Proceedings of the ICASSP, Orlando, vol I, pp 105–108
35. Povey D, Kanevsky D, Kingsbury B, Ramabhadran B, Saon G, Visweswariah K (2008) Boosted MMI for model and feature space discriminative training. In: Proceedings of the ICASSP, Las Vegas, pp 4057–4060
36. Povey D, Burget L, Agarwal M, Akyazi P, Feng K, Ghoshal A, Glembek O, Goel NK, Karafiat M, Rastrow A, Rose RC, Schwarz P, Thomas S (2010) Subspace Gaussian mixture models for speech recognition. In: Proceedings of the ICASSP, Dallas, pp 4330–4333
37. Reichl W, Chou W (1999) A unified approach of incorporating general features in decsion tree based acoustic modeling. In: Proceedings of the ICASSP, Phoenix
38. Reynolds D, Quatieri T, Dunn R (2000) Speaker verification using adapted Gaussian mixture models. Digit Signal Process 10:19–41
39. Sainath T, Kingsbury B, Ramabhadran B (2010) Auto-encode bottleneck features using deep belief networks. In: Proceedings of the ICASSP 2012, Kyoto
40. Saon G, Chien JT (2011) Bayesian sensing hidden Markov models for speech recognition. In: Proceedings of ICASSP, Prague, pp 5056–5059
41. Saon G, Chien JT (2011) Discriminative training for Bayesian sensing hidden Markov models. In: Proceedings of ICASSP, Prague, pp 5316–5319
42. Saon G, Chien JT (2011) Some properties of Bayesian sensing hidden Markov models. In: Proceedings of IEEE ASRU, Waikoloa
43. Saon G, Soltau H, Chaudhari U, Chu S, Kingsbury B, Kuo HK, Mangu L, Povey D (2010) The IBM 2008 GALE Arabic speech transcription system. In: Proceedings of the ICASSP, Dallas, pp 4378–4381
44. Schwenk H (2007) Continuous space language models. Comput Speech Lang 21(3). doi:http://dx.doi.org/10.1016/j.csl.2006.09.003
45. Soltau H, Waibel A (2000) Phone dependent modeling of hyperarticulated effects. In: Proceedings of the ICSLP, Beijing
46. Soltau H, Saon G, Kingsbury B, Kuo HKJ, Mangu L, Povey D, Emami A (2009) Advances in Arabic speech transcription at IBM under the DARPA GALE program. IEEE Trans Audio Speech Lang Process 17(5):884–894
47. Soltau H, Saon G, Kingsbury B (2010) The IBM Attila speech recognition toolkit. In: Proceedings of the IEEE workshop on spoken language technology, Berkeley
48. Stolcke A (1998) Entropy-based pruning of backoff language models. In: Proceedings of the DARPA broadcast news transcription and understanding workshop, Lansdowne, pp 270–274
49. Stolcke A (2002) SRILM – an extensible language modeling toolkit. In: Proceedings of the ICSLP, Denver, pp 901–904
50. Stolcke A, Bratt H, Butzberger J, Franco H, Gadde VRR, Plauche M, Richey C, Shriberg E, Sonmez K, Weng F, Zheng J (2000) The SRI March 2000 Hub-5 conversational speech transcription system. In: Proceedings of the NIST speech transcription workshop, College Park
51. Vergyri D, Kirchhoff K, Gadde R, Stolcke A, Zheng J (2005) Development of a conversational telephone speech recognizer for Levantine Arabic. In: Interspeech, Lisbon