Chapter 5
Language Modeling

Ilana Heintz

5.1 Introduction

The goal of language modeling is to accurately predict the next word in a sequence of words produced in a natural language. The history, or context, that is used to make that prediction might be long or short, knowledge-rich or knowledge-poor. We may base a prediction only on a single preceding word, or potentially using knowledge of all words from the start of the passage preceding the word in question. Knowledge-rich models can incorporate information about morphology, syntax or semantics to inform the prediction of the next word, whereas knowledge-poor models will rely solely on the words as they appear in the text, without pre-processing or normalization of any kind. This chapter describes knowledge-rich and knowledge-poor language models and how each might be useful in modeling Semitic languages.

Every language modeling technique and application must handle the issue of data sparsity: with limited amounts of data available on which to train a model, many parameters will be poorly estimated. In a model that estimates probabilities for two-word sequences (bigrams), it is unclear whether a given bigram has a count of zero because it is never a valid sequence in the language, or if it was only by chance not included in the subset of language expressed in the training data. As the length of the modeled sequences grows more complex, this sparsity issue also grows. Of all possible combinations of 4-grams in a language, very few are likely to appear at all in a given text, and even fewer will repeat often enough to provide reliable frequency statistics. The same is true of additional information (e.g., morphological or syntactic tags) added to each term in the sequence; the probability of encountering a given feature combination decreases as the complexity of the features increases. Therefore, if the goal of language modeling is to predict the

I. Heintz (✉)
Speech, Language and Multimedia, Raytheon BBN Technologies, Cambridge, MA, USA
e-mail: iheintz@bbn.com

I. Zitouni (ed.), *Natural Language Processing of Semitic Languages*,
Theory and Applications of Natural Language Processing,
DOI 10.1007/978-3-642-45358-8_5, © Springer-Verlag Berlin Heidelberg 2014

next word, the challenge is to find appropriate, reliable estimates of word sequence probabilities to enable the prediction. Approaches to this challenge are three-fold: smoothing techniques are used to offset zero-probability sequences and spread probability mass across a model; enhanced modeling techniques that incorporate machine learning or complex algorithms are used to create models that can best incorporate additional linguistic information; and particularly for Semitic language modeling, morphological information is extracted and provided to the models in place of or in addition to lexical information.

The data sparsity issue is particularly intense for Semitic languages due to their morphological richness, as is described earlier in this volume. The copious use of affixational morphology results in a great increase in vocabulary size. For example, the English present-tense verb "run" has the correct agreement as the first- and second-person singular, and first-, second-, and third-person plural forms. The same word form is also a singular noun. In the Semitic languages, different affixes would be used to create varying word forms indicating the correct person, number, and gender information, and the root-and-pattern morphology would be employed to derive a different form for the noun, as well as different stems to account for changes in tense or aspect. In order to build reliable language models, this increase in vocabulary size must be countered by collecting more training data, by applying morphological stemming techniques, or by employing sophisticated modeling techniques that better handle sparse data.

Language models are used in a variety of natural language processing applications. In automatic speech recognition, language models can be used to help choose the best sequence of words from an acoustic lattice. Similarly in machine translation, language models help to indicate the best path through a lattice output by a translation model. Natural-sounding phrases and sentences are produced by natural language generation with the aid of language models, as well. Discussions of Semitic language modeling occur predominantly in the automatic speech recognition literature, with some mention also in studies on statistical machine translation. These studies will be cited throughout this chapter.

This chapter proceeds by describing the utility of the perplexity evaluation of language models in Sect. 5.2. This is followed by an introduction to n-gram language modeling in Sect. 5.3, and an explanation of smoothing methods in Sect. 5.4. Understanding smoothing is necessary to understanding variations on n-gram language models that are described in Sect. 5.5. Section 5.6 comprises a review of how morphological information is incorporated into a number of Semitic language models. Section 5.7 summarizes and concludes the chapter with suggestions for the most useful approaches to Semitic language modeling.

5.2 Evaluating Language Models with Perplexity

Outside of any particular application, we can compare language models by comparing the likelihood that they assign to a previously unseen sequence of words. For a given language model (LM), we might wish to calculate:

$$P_{LM}(w_1 \dots w_n) \tag{5.1}$$

This probability is an element of the cross-entropy of the text:

$$H(P_{LM}) = -\frac{1}{n} log P_{LM}(w_1 \dots w_n) \tag{5.2}$$

As discussed in [36] among many other texts, cross-entropy is a measurement of the amount of uncertainty in a probability distribution. We usually calculate the probability of a sequence of terms as the sum of the probability of each term given its preceding context:

$$H(P_{LM}) = -\frac{1}{n} \sum_{i=1}^{n} log P_{LM}(w_i | w_1 \dots w_{i-1}) \tag{5.3}$$

It is customary to measure the *perplexity* of a test set given the language model, which gives the number of bits required to encode that text using the LM:

$$PP = 2^{H(P_{LM})} \tag{5.4}$$

A language model that encodes a test set most efficiently will do so with the *smallest* number of bits. Therefore, we optimize a language model by minimizing the perplexity of a given test set. Note that in modeling Semitic languages, the vocabulary of the train and test set is often changed to account for morphological properties of the language; in these cases, a variation on perplexity must be used to compare models fairly. Section 5.3 discusses these changes to the evaluation.

Equations (5.3) and (5.4) assume that we use the entire preceding context to calculate the probability of each term. Because that calculation is not feasible, we replace the $P_{LM}(w_i | w_1 \dots w_{i-1})$ term with an approximation, such as the probability of only the previous N terms, or of the previous N terms and appropriate features.

Perplexity is an often-cited measure of the strength of a language model. It is a measure of that model's complexity, and serves to quantify the entropy of the model in relation to an unseen text. Perplexity is both useful and limited because it is a measurement of a model that is intended for use in a larger system. A language model is normally a means to an end: speech recognition (ASR), machine translation (MT), natural language generation (NLG), and other natural language processing tasks all use language models in one or more steps of processing. Building complete systems for these tasks is usually more difficult and time-consuming than building the language model alone. Even with a stable system, testing whether a new language model improves that system's results can take a long time and consume computational resources. In contrast, perplexity is simple to calculate and easy to compare cross-model. Language models can be directly compared by perplexity without building acoustic models, translation models, or other required

system components. However, the measurement tells us little if anything about how well the model will perform *in context*. Perplexity indicates how likely the correct word is to follow a sequence of known words, but that likelihood usually ignores features like acoustic confusability, semantic confusability, and syntactic confusability, which will be the main hurdles in the tasks of ASR, MT, and NLG respectively. Perplexity can overestimate the difficulty of choosing the right word in some contexts, while underestimating the difficulty in others. If a particular difficult context shows up frequently, then improvements in perplexity may turn out to be misleading.

Most of the Semitic NLP studies cited in this chapter apply language models to the speech recognition problem. The challenge considered in this application is acoustic confusability, which is amplified in Modern Standard Arabic (MSA) by the concatenative morphology of affixes. Two words that differ only in a short prefix or suffix will be acoustically confusable, and there are many such word pairs in MSA. A language model that incorporates syntactic or morphological information may give differing probabilities to these words, and the reduction of perplexity may indicate such differences. But, we do not know whether the acoustic confusability problem is overcome unless the use of the language model in an ASR task results in a reduction of word error rate.

In summary, due to its ease of calculation and relationship to the well-understood property of cross-entropy, perplexity evaluations are often cited as indicators of language modeling utility. In these cases, care must be taken to assure that models are directly comparable in terms of their vocabulary and the vocabulary of the test set. Incorporating the language models into a complete NLP system and evaluating in situ, when possible, is a truer indicator of the models' utility.

5.3 N-Gram Language Modeling

A very common, basic form of language modeling is *n-gram language modeling*. A probability is assigned to each unit of language based on its frequency in a training corpus. A "unit of language" refers to some sequence of tokens, usually words. The order n of the language model can be single words (unigrams), or sequences of two, three, four, or more words (bigrams, trigrams, 4-grams, etc.). The simplest example of an n-gram model is a maximum likelihood model over unigrams, where each word is assigned a probability based on its count in a training corpus: $P(w) = \frac{count(w)}{\sum_w count(w)}$. A maximum likelihood bigram model takes into account the previous word: $P(w_i|w_{i-1}) = \frac{count(w_{i-1}w_i)}{\sum_w count(w_{i-1}w)}$. In theory, increased context leads to greater accuracy. A perfect theoretical model would base the prediction of each word on all of the preceding context: $P(w_i|w_1 \ldots w_{i-1})$. In practice, long n-grams do not repeat often enough to provide reliable statistics, so we estimate an imperfect n-gram model by restricting the context to only a few preceding words.

Calculating the perplexity of a test text given a bigram model only requires the specification of Eq. (5.4) to the bigram case:

$$PP_{bigram}(w_1 \ldots w_n) = 2^{-\frac{1}{n}\sum_{i=2}^{n} logP(w_i|w_{i-1})} \tag{5.5}$$

Similar formulas are derived for models with larger-order n-grams.

In using n-gram language modeling for Semitic languages, the definition of what constitutes a word or token in the model must be carefully considered. Often, normalization and tokenization tools are applied to the text before the language model is calculated. Normalization refers to reducing orthographic variation, such as changing the various forms of Arabic hamza to a single form. Tokenization methods such as stemming are used to reduce the size of the vocabulary by separating clitics and morphological affixes from stems. Reducing the size of the vocabulary through normalization and tokenization has the effect of increasing the frequency of each token, which in turn increases the reliability of the model. Normalization and tokenization also reduce the number of out-of-vocabulary tokens - those tokens that appear in the test text but not in the training text, and therefore may receive a zero probability. If a language model uses tokens that are smaller than words, then a post-processing step must be performed after decoding to return the tokens to their original state. This is called de-tokenization. El Kholy and Habash [20] and Al-Haj and Lavie [3] are recent studies that compare various types of normalization, tokenization, and the reverse processes in an English-Arabic machine translation application. Both studies find that a coarse level of tokenization, such as using surface-level segmentation as opposed to a fine-grained morphological segmentation, is sufficient to create useful language models, and also results in more accurate de-tokenization.

These studies and others cited below change the vocabulary of the language model by using morphemes or multiple words as the modeling units. Changing the vocabulary in this way results in a situation where comparing perplexity values is not mathematically sound; the conditions may have changed in such a way as to bias the calculation, and the perplexity results are no longer fully interpretable. Kirchhoff et al. [33] introduces a variation on the perplexity calculation that overcomes this problem by using a consistent normalization factor across models:

$$ModPP_{bigram}(w_i \ldots w_n) = 2^{-\frac{1}{n}\sum_{i=2}^{m} logP(w_i|w_{i-1})} \tag{5.6}$$

In this formulation, the normalization factor $\frac{1}{n}$ always counts the number of *words* in the test set. The number of tokens m may change from model to model based on tokenization methods, but the number of words n remains the same for a given test text. The average negative log probability of the test text is calculated by removing the exponent, as in [63] and [27]. These variations on the perplexity formula allow for a more reliable comparison of models that use differing vocabularies.

5.4 Smoothing: Discounting, Backoff, and Interpolation

Section 5.3 refers to the use of normalization and tokenization to counter the problem of out-of-vocabulary (OOV) terms, and the use of smaller contexts to increase the reliability of n-gram counts. This section discusses mathematical approaches to solving the problems of OOV terms and unreliable counts. These concepts are a necessary precursor to understanding the more complex types of LMs discussed in Sect. 5.5.

Smoothing refers to three related concepts: discounting, backoff, and interpolation. All three will be discussed in this section, which is modeled on the more complete discussion in [14]. **Discounting** refers to the movement of probability mass from frequent, well-modeled parameters to infrequent parameters with less evidence. The probabilities of all parameters in an LM must always sum to one, therefore discounting also implies normalization. **Backoff** is the term used to describe the use of lower-order n-gram probabilities when the evidence for higher-order n-grams is lacking. Higher-order (trigram, 4-gram) n-grams are preferred because they provide greater context for the predicted word, resulting in more accurate predictions. But this is only true when the higher-order n-grams exist in training text, and usually only if they exist with repetition. Because lower-order (unigram, bigram) parameters are more likely to repeat in a text, their estimates are more reliable, even if they provide less context. Backoff is used to counter the greater context of the high-order n-grams with the greater reliability of the low-order n-grams. Backoff is often done in a step-wise fashion: we use a lower-order probability where the higher-order probability is zero. **Interpolation** also takes into account higher-order and lower-order parameters, but in doing so creates a smoother distribution of probability mass than simple backoff. With interpolation, all of the higher-order n-gram probabilities are tempered by lower-order n-gram probabilities, regardless of their frequency.

5.4.1 Discounting

The simplest formulation for determining the probability of each n-gram requires no discounting; it is given by the maximum likelihood (ML) model:

$$P_{ML}(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i})}{\sum_{w_i} c(w_{i-n+1}^{i})} \tag{5.7}$$

Such an estimate results in zero-value probabilities for any unseen n-grams, an undesirable outcome, given that we always expect new words to occur in a text different from that used to build the model.

This outcome can be avoided by simply adding 1, or another constant, to the count of every n-gram. In a test condition, the unseen n-gram will be assigned

exactly this constant as its count. This has been empirically shown to be ineffective [22]. A more sophisticated technique is Good-Turing discounting, which involves calculating the count-of-counts: for every count r, how many n-grams appear r times? We calculate r^*, the Good-Turing discounting factor, by smoothing the count-of-counts:

$$r^* = (r+1)\frac{N_{r+1}}{N_r} \tag{5.8}$$

where r is the frequency of an n-gram type (its token count) and N_r is the number of n-grams (types) with that frequency. For an n-gram $w_i \dots w_{i-n+1}$, we count its occurrences r, then estimate its probability using its discounted count r^*:

$$P_{GT}(w_{i-n+1}^i) = \frac{r^*}{N} \tag{5.9}$$

The probability mass that is taken away via discounting is "given back" in the form of probability mass on unseen events: the total probability attributed to unseen objects is $\frac{N_1}{N}$, the maximum likelihood of all singleton n-grams [23].

To understand Eq. (5.8), imagine a corpus of N tokens. We remove one token from the corpus to create a single-token 'held-out set', then count the occurrences of that type in the remaining 'training' set. We find that token in the training corpus one fewer times than its actual occurrence in the full corpus. If we repeat the 'held-out' experiment for each token, we find that any token that appeared r times in the training set has a true count of $r+1$. Now consider a held-out token that does not appear in the training set. From the perspective of the training set, it is an unknown, but we know that it in fact has a singleton count in the true corpus. This is why we attribute approximately a singleton count to all unknown tokens. We find the total count of all singletons: $(0+1)N_1$, then divide that count by the number of unknowns N_0 (Eq. 5.8), assigning to each unknown token an equal portion of the total singleton count. We divide again by the token count of the training corpus to derive the appropriate probability mass for each unseen token (Eq. 5.9). The same logic is true when we consider the discounted value of singletons, twice-appearing types, and so on.

To calculate the Good-Turing discount, the training data must be diverse enough to preclude any zero counts r; in particular, there must be tokens with low r counts, including $r = 1$ (which is to say, we do not prune singleton tokens before estimating the Good-Turing probabilities). At the upper ranges of r, where zero values of N_r are likely to naturally occur, the values of N_r are interpolated using simple linear regression. This step removes the zero values, and the estimates reflect a trend of fewer items, rather than leveling off at $N_r = 1$. In general, this method of calculating r^* results in estimated counts that are slightly less than the actual count for low values of r, and close to the actual count for high values of r. Less probability mass is taken from high-frequency words (and occasionally probability mass is added to them via interpolation, due to the variability of N_r for high values of r) because

they are more trustworthy. We are more skeptical of the repetition of low-frequency words, therefore we take more probability mass from them to apply to unseen words.

With this estimated value of r^*, Eq. (5.9) can be used to calculate the estimated probability of each item in the training data. This discounting method is often used to estimate n-gram probabilities in conjunction with one of the backoff or interpolation methods discussed next.

5.4.2 Combining Discounting with Backoff

A very common form of combining backoff with discounting is known as Katz backoff, introduced in [30]. In this algorithm, frequent n-grams are not discounted; they are trustworthy and their estimates are not changed. Low-frequency n-grams, already considered unreliable, lose some of their probability mass to unseen n-grams. This discounting is done with a variation of the Good-Turing discount factor. Furthermore, backoff weights are used to ensure that the total probability mass of the model remains equal to 1. Formally, where $c(x)$ indicates the count of x and k is a frequency cutoff constant:

$$p_{Katz}(w_i \dots w_{i-n+1}) = \begin{cases} p_{ML}(w_{i-n+1}^i), & \text{if } c(w_{i-n+1}^i) > k \\ d_r(w_{i-n+1}^i), & \text{if } 0 < c(w_{i-n+1}^i) \leq k \\ \alpha(w_{i-n+1}^{i-1}) p_{ML}(w_{i-n+2}^i), & \text{if } c(w_{i-n+1}^i) = 0 \end{cases} \tag{5.10}$$

The discount factor d_r is a variation of the Good-Turing discount factor; the discount is modified so that the total probability mass of the model remains constant, despite the choice to *not* discount high-frequency n-grams. The backoff weight α is related to the discounting factor; in the case that we use backoff, α applies the probability mass gained from the discounting factor evenly across the n-grams that could result from the backoff n-gram. The exact calculation of d_r and α are specified in [30] and [14]. What is important to note is the close coordination of backing off in the third term with the discounting in the second term – together they are used to eliminate zero counts and obtain optimal reliability in the language model.

5.4.3 Interpolation

In addition to providing an algorithm for combining a well-motivated discounting factor with the benefits of backoff, the Katz algorithm also takes advantage of *interpolation*. In the last step, the probability of an n-gram is calculated iteratively by accounting for all relevant lower-order n-grams. Jelinek-Mercer smoothing [29]

also takes advantage of this recursive interpolation, but does not require calculating the Good-Turing discounting factor or backoff weights. It simply says that, for any n-gram, its probability should be estimated by taking into account the probability of the whole context, as well as the probability of all shorter contexts. This is again meant to balance the accuracy gained from the greater context of a high-order n-gram with the greater reliability of the more frequently seen low-order n-grams. Recursive interpolation is formulated as follows, where P_{ML} refers to the maximum likelihood probability [10, 14]:

$$P_{interp}(w_i|w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} P_{ML}(w_i|w_{i-n+1}^{i-1}) + (1-\lambda_{w_{i-n+1}^{i-1}}) P_{interp}(w_i|w_{i-n+2}^{i-1}) \tag{5.11}$$

The interpolation parameters λ determine how much each order n-gram will contribute to the total probability. The values of each $\lambda_{w_{i-n+1}^{i-1}}$ can be learned via the Baum-Welch algorithm, and may be calculated over word classes. Rosenfeld [52] states that these parameters need not be specified exactly; their variance within 5 % of the best value will create little difference in the perplexity of a held-out set.

Witten-Bell smoothing [6, 67] is another recursive algorithm that interpolates lower-order and higher-order n-gram counts. The key insight is the use of the count of unique word types w_i that follow a given prefix: $|w_i : c(w_{i-n+1}^{i}) > 0|$. This value is used to determine the values of λ in Eq. (5.11). To determine a particular $\lambda_{w_{i-n+1}^{i-1}}$, we divide the *type* count above by the *token* count of all n-grams with that prefix:

$$1 - \lambda_{w_{i-n+1}^{i-1}} = \frac{|w_i : c(w_{i-n+1}^{i}) > 0|}{|w_i : c(w_{i-n+1}^{i}) > 0| + \sum_{w_i} c(w_{i-n+1}^{i})} \tag{5.12}$$

This calculation is intended to answer the question, "How likely are we to see a new unigram following this prefix?" The answer is used in determining the amount of smoothing at each step of interpolation.

Absolute discounting is another form of interpolation. Ney and Essen [44] and Ney et al. [45] show that using the same discounting factor for all n-grams, regardless of order or frequency, can be effective if that discounting factor is properly set. For instance, an empirical study [45] shows the following estimate to be an effective constant for absolute discounting:

$$Discount = \frac{n_1}{n_1 + 2n_2} \tag{5.13}$$

where n_1 and n_2 indicate the number of singleton and two-count n-grams in the corpus, respectively. Again, the algorithm is recursive so that all lower-order n-grams contribute probability mass to an n-gram. Absolute discounting led to the widely-used Kneser-Ney discounting [35]. In this algorithm, the concern is over how many prefixes a given unigram follows. Put another way, how likely are we to find a given word following a new prefix? For a given unigram, its probability mass calculated over all prefixes will be equal to its maximum likelihood probability:

$$\sum_{w_i-1} p_{KN}(w_{i-1}w_i) = \frac{c(w_i)}{\sum_{w_i} c(w_i)} \tag{5.14}$$

This constraint is incorporated into a model that includes an absolute discounting factor and recursion over lower-order backoff models. The modified version proposed by Chen and Goodman [14] is a fully interpolated model – all higher-order n-gram probabilities are discounted and interpolated with lower-order probabilities. The formula for modified Kneser-Ney smoothing, which takes into account both the intuition of [35] regarding known n-gram prefixes and the intuition of [29] regarding interpolation, is given in [14], along with its derivation and further motivation.

In their careful examination of many variations of the smoothing algorithms mentioned, [14] find that their modified Kneser-Ney algorithm is the most successful at reducing the perplexity of a test set. This is due to its use of an absolute discounting factor *and* interpolation of lower-order models over all frequencies of n-grams.

These smoothing methods are not particular to one language or another. They are each designed to help overcome the sparse data problem, to give a non-zero probability to unseen n-grams, and to allow the probabilities of the component n-grams to be distributed more evenly across all parameters of the model. They are used in combination with varying types of language models that incorporate information about semantics and syntax, as described in Sect. 5.5. Several of the models described in Sect. 5.5 also attempt to overcome some limitations of these smoothing algorithms.

5.5 Extensions to N-Gram Language Modeling

While the most widely-used language model is the simple n-gram model, there are many variations on the theme. These variations incorporate longer-distance dependencies, syntactic information, and semantic information. The following sections will briefly describe the seminal research for each type of model and its existing or potential application to Semitic natural language processing.

5.5.1 Skip N-Grams and FlexGrams

The use of n-gram modeling is pervasive in natural language processing because of its simplicity and surprising effectiveness. Despite the utility of a word's immediate history in predicting the probability of that word, there are many cases where a slightly different history – not necessarily a longer history, but a different one – may be more effective. For instance, consider the phrase "ravenous lions, voracious tigers, and famished bears." The appearance of the word *bears* is best predicted by its two predecessors *lions* and *tigers*, but a trigram model will only be privy to the

context *and famished*. One solution to this issue is a skip-gram model, described in [24]. First, the usual collection of n-grams is defined. Then, the parameters are expanded by collecting alternative histories for each word: all bigrams and trigrams within the sentence that skip over k tokens preceding it. The number of n-grams for a given sentence greatly increases as the number of skips allowed is increased. An evaluation that centers on the coverage, rather than the perplexity, of a test text is useful in this application. Indeed, coverage increases as the number of skips k increases. When the domain of the train and test are mismatched, e.g., broadcast news as training data and conversational speech in the test set, the coverage does not improve as significantly. This shows that the skip trigrams are not over-generating contexts to the point of false positives. A second evaluation shows that the use of skip-grams can be as effective, or more so, as increasing the size of the corpus. This result may be especially relevant for modeling low-resource Semitic languages such as Amharic, Maltese, and Syriac.

Similarly, [69] describe the use of *flexgrams* to overcome the limited context of n-grams. In this method, the history may arise from anywhere in the sentence preceding the predicted word. This method is combined with morphological decomposition to better model the Turkish language. A reduction in perplexity is shown with the use of flexgrams and morphological information. Therefore, this may be a method that is also applicable to the morphologically rich Semitic languages.

5.5.2 Variable-Length Language Models

When the required storage space and computation time of a language model need to be minimized, variable-length language models are useful. To create the optimal model, only the most useful parameters are kept. This can be achieved in a number of ways, including but not limited to the following techniques:

1. Discard longer n-grams if they do not contribute to the model. Kneser [34] shows that longer n-grams can be discarded if they do not meet a frequency threshold, e.g. singletons n-grams, or if they do not reduce the distance between the current LM and the optimal (full) LM. Kneser [34] provides an algorithm for optimizing the pruning function in a computationally tractable way. Seymore and Rosenfeld [57] compare the log probability of the full and reduced model having removed each n-gram in turn, pruning those that do not increase the probability by a threshhold amount. Similarly, the method in [62] prunes the LM by measuring the relative entropy of the pruned and original models, where the perplexity of the original training set is used to express entropy. Any n-gram that raises the perplexity by less than a threshhold is removed, and backoff weights of the remaining n-grams are re-calculated. Siivola [61] also perform re-calculation of backoff weights and discount parameters after n-grams are removed through pruning, always retaining a model that is properly smoothed according to Kneser-Ney smoothing. This pruning algorithm has been tested

successfully on morphologically rich languages such as Finnish, especially when morphemes are used as the building units for the language model [60]. The method was not as successful for Egyptian Arabic data, however this may have been due to the small size of the corpus. The consistent result in these studies is that the outcomes of applying pruned models vary only slightly from outcomes using larger models, so that time and storage savings are achieved without sacrificing too much accuracy, either in perplexity or application results. Despite the growing use of Kneser-Ney smoothing, [13] show that when aggressive pruning is performed, for instance pruning an LM to a mere 0.1 % of its original size, Katz smoothing is a better choice.

2. Niesler and Woodland [47] take the opposite approach by adding more context to shorter n-grams if they do contribute to the model. In [51], the variable-length n-grams are modeled as a type of Probabilistic Suffix Tree. The branch of a tree can grow only if the resulting model retains a Kullback-Liebler distance less than some threshold from the optimal, full model. Otherwise, the smaller (fewer parameters) model is used. This approach is also used in [61], where a consistent modification of backoff and smoothing parameters are applied to ensure that as the model grows, it has proper Kneser-Ney smoothing at every step. Again, similar task results can be achieved using a more compact language model that excludes low-impact n-grams.
3. A different approach is that of [18], which changes the very construction of an n-gram. Rather than assuming that each unit of an n-gram is a single word token, instead the text is first segmented using the EM algorithm to choose the most informative segments. Collocations, for instance, can be grouped as a single token. This reduces the number of unigrams, creating a smaller model. Perplexity must be calculated over a text that has been segmented in the same fashion.

The benefits of each of these methods is that they provide a more theoretically appropriate model than a fixed-length n-gram model, as words in a sequence are dependent on different length histories. Also, the training algorithms are such that only as many parameters as are useful to the model are used, and the model size can be tuned on held-out data. This is a storage savings for models that would otherwise have size exponential in the order of the n-gram. This is perhaps less appealing when dealing with languages that lack in data, but more appealing for languages with explosive vocabulary growth, such as Modern Standard Arabic. The intuitions that inspired variable length n-gram modeling are surely applicable to language modeling in the Semitic language domain. Despite the lack of literature that applies these models to Arabic or other Semitic languages, it is an area that will likely produce gains in language model accuracy and utility for those languages. In particular, the third approach could be applied after morphological segmentation to reduce the OOV rate and supply the most reliable statistics to the model. The techniques mentioned above could also be applied to morpheme-based n-gram models. Even so, class-based modeling, described in the next section, may be more useful than variable-length modeling for handling large vocabulary

applications; [47] combines class-based modeling with variable-length modeling to positive effect.

5.5.3 Class-Based Language Models

One way to reduce the sparsity issue in n-gram modeling is to assign each word to a class. The class may be based on semantic or syntactic principles. If an unknown word can be successfully assigned to a class, then it is easy to apply the statistics of distribution associated with that class to the unknown word. Brown et al. [10] introduced class-based models with the following premise: "If we can successfully assign words to classes, it may be possible to make more reasonable predictions for histories that we have not previously seen by assuming that they are similar to other histories that we have seen." Where c_i represents the class that word w_i is mapped into:

$$p(w_i|w_{i-n+1}^{i-1}) = P(w_i|c_i)P(c_i|c_{i-n+1}^{i-1}) \tag{5.15}$$

Alternatively, one might sum over the classes c_i if a word is seen as having a distribution over classes. Rather than depend on the token history of each word, we instead calculate the probability of the word given its class and the class history of that word. The intention is to have histories that have been seen more often and are therefore more reliable. There are many ways that the word classes can be derived. In [10], a point-wise mutual information-based measure is used to group words into semantic classes. These classes and the resulting language model can be used in interpolation with word-based models.

As for Semitic language modeling literature, [70] use class-based modeling for an Arabic language application. Classes are used in back-off: when w_{i-n+1}^{i} is unknown, the model backs off to $w_{i-n+2}^{i}c(w_{i-n+1})$. Alternatively, the authors create a hierarchy of these back-off possibilities, training a language model for each level of the tree, and linearly interpolating the models. This allows the model to incorporate statistics from the most accurate n-grams and the most reliable class-based statistics.

In [32], class-based language models are developed based on Arabic morphological classes and using word clustering algorithms. These class-based models are combined in a log-linear fashion with stem-based and root-based morpheme models and with two word-based models. The parameters of the log-linear model are derived automatically, and the analysis shows that the morpheme-class model is among the more influential of the group, obtaining a high weight for interpolation. In an ASR application, both of the class-based models bring down the word error rate when combined with the word model, as compared to the word model alone. This shows that interpolating class-based models with more fine-grained information can lead to lower perplexity *and* improved ASR scores.

The derivation of the class-based models and a more careful comparison to other models, including factored language models, is explored in [33], a study on Egyptian Colloquial Arabic speech recognition. Here, class-based models are defined by morphological components: stems, morphs, roots, and patterns. The models associated with each class are defined and calculated separately and then interpolated. Kirchhoff et al. [33] uses a slightly different formulation of the class-based language model than [10]:

$$P(w_i|w_{i-1}) = P(w_i|c_i)P(c_i|c_{i-1})P(c_{i-1}|w_{i-1}) \tag{5.16}$$

The class-based models fare well in the ensuing analysis, especially when combined with stream models. Stream models replace each word with a specific morphological component and derive an n-gram model over that morpheme stream. This shows that for Egyptian Colloquial Arabic, very high-level and very low-level information can be successfully combined to improve ASR scores.

5.5.4 Factored Language Models

Kirchhoff et al. [33] introduce Factored Language Models (FLMs) with an eye towards Arabic and other Semitic languages. FLMs take into account as many varied kinds of morphological, class, or semantic information as is available for each word. This is an extended form of n-gram modeling where the backoff procedure is more complex than the standard model. Typically, for a trigram model, in the case that a particular trigram is not among the parameters, the model will back off to the most appropriate bigram by ignoring the most distant word and applying a backoff weight [30]. In an FLM, rather than drop the most distant word, instead we consider its class, part-of-speech tag, or other feature for which a reliable weight has been derived. FLMs are formulated as follows for a trigram model, similarly for higher-order or bigram models:

$$p(f_1^{1:K}, f_2^{1:K}, \ldots, f_N^{1:K}) \approx \prod_{i=3}^{N} p(f_i^{1:K}|f_{i-1}^{1:K} f_{i-2}^{1:K}) \tag{5.17}$$

where $1:K$ represent the K features annotating each of the N words. Modeling is done over f, a group of factors, rather than a word w. When an n-gram of complete factors is not specified in the model, backoff can proceed along many routes, dropping any of the factors for which there is an appropriate backoff weight in the model. Many different backoff *paths* can be taken, and different discounting models integrated along those paths. For instance, one might first drop the most specific information in a bundle, the word itself, and use Kneser-Ney discounting to do so. If the n-gram is still not found, one might drop a syntactic tag from the bundle, and use Good-Turing discounting at this juncture. The choice of backoff

can be determined a priori if faith in the features and linguistic knowledge is sufficient to choose a reliable path. However, given the number of possible paths, it is recommended to use an automatic method of choosing or calculating the path. In [33], the authors use generalized parallel backoff, which takes into account all of the possible backoff paths via averaging, multiplication, or a smooth probability distribution.

Alternatively, backoff paths can be chosen via genetic algorithms, as introduced for use with FLMs in [19]. These are the preferred method, as they are able to take into account many if not all of the possible backoff paths, choosing the best in a motivated fashion: the genetic algorithms are trained using perplexity of development data as the optimization criterion. The automatically chosen paths produce a lower perplexity on a test set than n-gram models, hand-chosen backoff models, and random models (but are not compared to generalized parallel backoff).

The benefit of FLMs is the ability to use many morphological and syntactic characteristics of each word; these properties are plentiful in Semitic languages when counting affixes, stems, roots, and other possible morphological categories. The same features that give Semitic languages their ever-expanding vocabularies and intricate complexity are those that can be used to make a more informed language model, without suffering the sparsity problem. On the other hand, one must have tools to produce each of these features, and such tools (especially root finders) are not always available for the lesser-studied languages and dialects. It is sometimes possible to adapt tools from one language to another if there is a tolerance for error; it is unclear whether Factored Language Models are robust to such tagging errors. Given the genetic algorithms, it may be the case that they are robust, as the unreliable features can be dropped early in the backoff path if they do not positively influence the perplexity of the development set (or other training criterion).

5.5.5 Neural Network Language Models

Schwenk [55] describes a method of obtaining language model probabilities in a continuous space using neural networks. An input layer, two hidden layers, and an output layer comprise the neural network. The input layer accepts 4-grams extracted from a corpus. The first hidden layer projects this vocabulary of possibly hundreds of thousands of terms into a smaller space of only 50–300 dimensions. The second hidden layer is typical of neural network algorithms: its weights are trained using non-linear back-propagation. The output nodes represent each word in the vocabulary. If the input is a set of words representing a three-word history, then the output layer contains the probability that each vocabulary word follows that history. In training, the optimization criterion is the maximization of the log-likelihood of development data (or minimization of the perplexity, equivalently). These outputs are interpretable as posterior probabilities when the softmax normalization algorithm is applied.

Training neural networks is more time-consuming than estimating a simpler Katz or Witten-Bell language model. To speed training, the output layer may be reduced to only the most frequent words. Tuning the probabilities of these frequent words via neural networks is useful simply because they will appear often in the training data; reducing word error rate on these nodes will more drastically reduce the overall word error rate than will reducing word error rate on rare words. To obtain the necessary coverage of the vocabulary, the neural network models are interpolated with traditional backoff language models.

The estimation of language model probabilities is done in a *continuous* or *distributed* space when using neural networks, as opposed to the discrete calculations performed for traditional back-off models. Continuous probability estimation is better understood, less ad-hoc, and presumably more accurate than discrete models. Neural network models expand linearly with the size of the vocabulary, rather than exponentially as is typical of traditional backoff modeling.

Schwenk [55] describes additional methods to speed training, such as allowing multiple examples to be given as input at each training epoch, and randomly sampling the data from multiple corpora in order to achieve the best adaptation for different language styles. Training over large corpora can also be achieved without great computational demands by using this randomized sampling method. Larger n-grams can be modeled with little effect on training time, however the data sparsity that is attendant with such models is not diminished by using the neural network method.

These methods work well for lattice rescoring with state-of-the-art speech recognition systems in English, French, and Spanish on texts ranging in style: broadcast news, meetings, and conversational data.

Emami et al. [21] apply this method to Arabic data. A variety of morphological data is included in the input layer to create a richer representation of the context, without multiplying the number of parameters and requiring a more complex training technique, as is the case with Factored Language Models. Maximum entropy part-of-speech (POS) tagging, maximum entropy diacritic restoration, and segmentation of words into multiple affixes and stem using a cascaded weighted finite state transducer are the tools used to provide morphologically informed features. These properties are modeled as features of the word histories that are the input to the neural network algorithm. The features are simply concatenated to represent each word as a collection of features. This increases the model size only linearly with the number of features, due to the projection of the input space to a more manageable dimensionality in the first hidden layer of the neural network. Perplexity decreases as more informative features are added to the input, and when the models of varying input types are interpolated. To calculate perplexities, the small-vocabulary neural network model is interpolated with a traditional backoff model with the full vocabulary; the backoff model probabilities are used whenever the neural network model does not have coverage of a word.

Despite the reductions in perplexity, [21] do not show an improvement in word error rate when morphologically-rich neural network language models are used to rescore ASR lattices. This may well have been due to errors in the POS

tagging, segmenting, or diacritic restoration of the recognized word lattices. The work is expanded in [38], where a slightly different set of features are employed: along with segmentation and POS tags, shallow-parse chunk labels and full-parse headwords both preceding and following the predicted word are used as features. The probabilities produced by the neural network language model are used to rescore an N-best list of ASR outputs in place of lattice rescoring. In this study, word error rate did decrease by significant amounts on both broadcast news and conversational data, especially when the more complex syntactic features – previous and following exposed headwords – were included in the neural network language model.

Therefore, the incorporation of morphological and syntactic features into a continuous-space language model has been shown to be beneficial to Semitic natural language processing, in particular in the speech domain.

5.5.6 Syntactic or Structured Language Models

Chelba and Jelinek [12] describe a structured language model (SLM) in which a left-to-right parsing mechanism is used to build a model that can be applied in the first pass of speech decoding. The challenge in training and testing is to find the best equivalence class for the history of the current word; the n-gram history is often not sufficient or is misleading, so a syntactically-informed history is used to replace it.

The SLM builds the syntactic structure incrementally; the part-of-speech tags are used to predict the next word as well as that word's tag, the structure of that portion of the parse, and the non-terminal label on the headword. There are multiple dependencies between these parameters, all of which are learned from data in the Penn Treebank.

The model that results is similar to a skip-gram language model, where the number of words skipped in the history of a word depends on the context. Alternatively, the model can be described as a kind of linear interpolation of varied-order n-gram models. Many of the predictions made by this model are long-distance predictions that a trigram model does not capture.

As in other studies, the language models are tested in an ASR application and are found to bring down word error rate a small amount. The computational load of training and applying the SLM, however, are reported to be significant. The use of this model in conjunction with a complementary model that focuses on topical or semantic modeling may be rewarding. Section 5.5.8 discusses a study that does incorporate both syntactic and semantic knowledge.

The incorporation of parse trees into language modeling is, clearly, dependent on the availability of parse trees in that language. At present, this resource and its attendant automatic parsers are available for Modern Standard Arabic, but are in short supply for other Semitic languages or dialects of Arabic. When the reliability of automatic parses is of sufficient quality to merit their use in other tools, then structured language modeling will undoubtedly be useful for Semitic languages.

5.5.7 Tree-Based Language Models

Bahl et al. [5] introduce tree-based language models in the context of speech recognition. In this computation-heavy algorithm, binary trees are constructed such that at each node a yes/no question is asked. Each question regards whether some predictor variable, such as the previous word, nth previous word, previous headword, etc., belongs to a given class of words. The classes are also syntactically defined, including nouns, verbs, and plurals. At the leaves of this tree are sets of predicted words and their probability distribution. An automatic set-building algorithm generates a good, if not optimal, model. The probabilities on the leaves are estimated by taking the average entropy of all of the nodes leading to that leaf; that is, the average conditional entropy of the sets of words defined by the predictors at each node.

The resulting models have lower perplexity than trigram models and would seem to be good complements to them in a log-linear or other combination scheme.

This type of model has not been widely applied, perhaps due to the intensive computing described in the 1989 article. With more modern computing resources, such a model would be less onerous to produce, and could handle a larger vocabulary than 10K words. Future research could show that tree-based language models form a useful complement to the other language model types described in this chapter. The types of features used are perhaps more easily produced than those used in, e.g., FLMs, therefore this might be an appropriate technique to use with resource-poor languages like Maltese and Syriac. Also, that the model improves results on a small-data application may point to its utility for the same languages.

5.5.8 Maximum-Entropy Language Models

Maximum Entropy (MaxEnt) models are used to incorporate varying kinds of information in a single language model. They are designed to preserve all uncertainty except where the data explicitly provides evidence for a choice. As described in [8] and elsewhere, the MaxEnt formula takes the form:

$$P_{\Lambda}(y|x) = \frac{1}{Z(x)} exp(\sum_{i=1}^{F} \lambda_i f_i(x, y)) \tag{5.18}$$

The values of $\lambda_i \in \Lambda$ for each feature i are calculated over the training set, where every data point x is described by F features and a class label y. Often (but not always), the features are binary, so we can describe a certain feature f_i as *firing* when the feature is true and variable x belongs to class y. The general nature of this algorithm allows the features to encompass many kinds of information. The normalizing factor Z assures that the resulting probabilities (x, y) are a proper probability distribution:

$$Z(x) = \sum_{y \in Y} exp(\sum_{i=1}^{F} \lambda_i f_i(x, y)) \tag{5.19}$$

Rosenfeld [52] describes the benefits of using MaxEnt models as well as a step-by-step derivation of the basic training algorithm. In that study, maximum entropy modeling is used to interpolate long-distance functions such as trigger words and skip n-grams together with a baseline trigram model. Mutual information calculations are used to develop trigger models – pairs of words in which each word is informative about the other word's presence in a document. Skip n-grams form a good counterpart to traditional n-grams, incorporating more knowledge about a word's history without necessarily incurring the same data sparsity effects of a higher-order model. These kinds of models can be linearly interpolated, which has the benefit of ease of calculation. But, the theoretical grounds of each model are lost; while each model is initially parameterized at a local level, the weights in the interpolation algorithm (trained via EM algorithm) are parameterized on a global level. MaxEnt learning can be used to combine these models without suffering this loss of theoretical grounding. The constraints of each model are relaxed in a way that allows them to interact, benefitting each other and strengthening the predictions. With maximum entropy modeling, the probability of a given bigram will change based on its context and on how functions from the various models handle that context. Experiments show that combining language models via MaxEnt is superior to linear interpolation. Not all of the long-distance models are as useful as hoped, but the interactions are interesting and in many cases complementary. The effect of the trigger model in particular is shown to match its theoretical benefit. The long-distance models are ideal for adaptation from one domain to another, provided that the adaptation training data is appropriate and that local-context models are also incorporated.

Rosenfeld [52] describes training the weights λ_i using the Generalized Iterative Scaling algorithm. Other, more effective models include Improved Iterative Scaling, as introduced in [8], and gradient ascent. Maximum entropy modeling is used in many tasks aside from language modeling; a review and comparison of these and other MaxEnt training techniques is given in [41].

Sethy et al. [56] and Chen et al. [15] describe a model that uses the MaxEnt algorithm together with class-based modeling to improve word error rates in ASR. The introduction of class histories helps to make the model more compact and helps to reduce the data sparsity problem that pervades n-gram modeling. Importantly, [15] show improved scores on an Arabic ASR task using this method.

Sarikaya et al. [54] also use Maximum Entropy modeling to combine varying features in a language model for an Iraqi Arabic ASR task. Word tokens are segmented into morphemes and decorated with part-of-speech and other morphological information. Maximum entropy modeling takes these features, designed in a tree structure, as features to train a morphologically-informed language model. Joint word and morpheme models are used to rescore word-based n-best lists and succeed in reducing word error rate. The same technique, combining morphological with

lexical information in a maximum entropy language model, is also used to get improved results in a statistical machine translation task between English and Iraqi Arabic in [53]. These studies show that maximum entropy modeling is a fruitful arena for combining different kinds of linguistic information, including the kinds of morphological information that have been shown to be useful to Semitic language modeling.

Khudanpur and Wu [31] describe a method for combining semantic and syntactic information in a maximum entropy language model. Semantic information is incorporated by building a topic-dependent language model; in contrast to other studies where a separate LM is built for each topic and interpolated with a full-vocabulary LM, [31] use maximum entropy modeling to create a single, fully-interpolated model with fewer parameters. Furthermore, they derive syntactic information with a trained parser, and incorporate the top N parses and probabilities into the same language model, again using a maximum entropy technique. The topic and parse data add more global information to the model, allowing it to make predictions with a history greater than only the two previous words. The combined model is successful in lowering word error rates on an English conversational speech ASR task. The resources required to replicate this are significant: to build the topic-based language model, [31] use a corpus with manually labeled topics (per conversation), and a well-trained parser is required for the syntactic information. In performing this task on Semitic data for which fewer resources are available, one might consider using an alternative method for assigning topics to training data, e.g. Bayesian topic modeling, and a simpler parsing methodology that incorporates long-range dependencies.

5.5.9 Discriminative Language Models

Discriminative language models are an alternative to n-gram models introduced by Roark et al. [50]. The discriminative model tries to solve the equation:

$$F(x) = \underset{y \epsilon GEN(x)}{\text{argmax}}\ \Phi(x, y) \cdot \bar{\alpha} \tag{5.20}$$

In this equation, x represents the input to a model; for instance, these inputs may be an acoustic signal of speech. The function $GEN(x)$ produces all of the possible outputs given this input, that is, all of the possible sequences of words given that acoustic input. The function $\Phi(x, y)$ is a set of feature values over x and y, and $\bar{\alpha}$ is a set of weights chosen so that, when the function produces the maximum value, y is the correct sequence of words spoken in the acoustic signal. The dual concerns of discriminative language modeling are finding an appropriate and effective feature vector Φ and an algorithm for training the weights $\bar{\alpha}$. As regards feature vectors, the technique used in [50] takes advantage of application-specific information to build the most effective models. Regarding the training of $\bar{\alpha}$, the authors show

how two discriminative models, the perceptron algorithm and the global conditional log-linear model (GCLM) can be successfully applied.

The steps of the process are:

1. Build a baseline recognizer with an acoustic model and n-gram language model that produces output in lattice format.
2. Use these baseline lattices to derive features for discriminative LM training, and as the training data for training the weights $\bar{\alpha}$.
3. Use the new language model parameters to refine the decoder output in a second pass over the lattices.

Therefore, the set of outputs $GEN(x)$ on which the discriminative model is trained are the sequences in the baseline lattices. There are two kinds of features that are derived from these lattices: the log probability of the best path y (either the correct path if it exists in the lattice, otherwise the minimum error path is found), and the frequency of each n-gram in y. Rather than estimate probabilities for all n-grams, only those n-grams found in the best path y are considered. This produces great savings in training and storage for the discriminative language model.

The perceptron algorithm increases or decreases the weights of $\bar{\alpha}$ at each iteration whenever the predicted output is incorrect, eventually guiding the algorithm to the maximum number of correct answers. In order for the training to converge, the data must be separable. This means that for each incorrect answer, its score is at least δ different than the score for the correct answer. The number of training iterations is dependent on the value of δ. To apply the perceptron algorithm for training, the weights on all paths in the baseline lattice are initialized to zero and are adjusted so that y receives the greatest weight. The final weights on the lattice become the parameters $\bar{\alpha}$. These parameters are used both for second-pass decoding, and as the feature set and starting point for the global conditional log-linear model (GCLM) training algorithm.

GCLMs, in contrast, are exponential, and assign a conditional probability to each member of $GEN(x)$.

$$p_{\bar{\alpha}}(y|x) = \frac{1}{Z(x, \bar{\alpha})} exp(\Phi(x, y) \cdot \bar{\alpha}) \tag{5.21}$$

The conditional distribution over the members of $GEN(x)$ are used to calculate the log probability of the training data, given parameters $\bar{\alpha}$. In GCLM, the weights on the language model represent probability distributions over the strings. The same features as mentioned above are used for training. The parameters of the model are estimated from the training data and applied as a language model in a second pass of decoding.

Both the perceptron and GCLM models are trained as weighted finite state automatons (WFSAs). The $\bar{\alpha}$ parameters are derived from the WFSA. The number of features used in training can be kept to a minimum by only considering those n-grams that appear in baseline output, which also results in only the most crucial

parameters being included in the model. The resulting language model can be directly combined with the baseline lattice for re-scoring.

Presumably, the same approach could be taken with machine translation and its lattices. The refined language models will then be configured to best discriminate between semantically, rather than acoustically, confusable terms.

Kuo et al. [37] introduce a third method of training the discriminative LM, the minimum Bayes risk calculation. In this case, the $\bar{\alpha}$ parameters of the model are trained by calculating the number of errors contained in each hypothesis. The loss associated with each hypothesis is compared to the same loss without a given feature included; if the loss is worse without the feature, then α_i associated with feature i is positively updated, otherwise it is negatively updated. Kuo et al. [37] show that this algorithm produces improved word error rate on ASR for Modern Standard Arabic, especially when morphological features are used to reduce the out-of-vocabulary rate. Additionally, the authors describe the Bayes risk metric, which allows them to train the discriminative LM in an unsupervised way. Instead of comparing each hypothesis to a reference transcription, each hypothesis is compared to every other hypothesis, and the sum of the differences represents the Bayes' risk metric:

$$L(y|x_i) = \sum_{y' \in GEN(x_i)} L(y, y')p(y'|x_i) \tag{5.22}$$

where $L(y, y')$ represents the number of errors in hypothesis y as opposed to y', and $p(y|x_i)$ is the posterior probability of hypothesis y. The hypotheses and probabilities are given by a baseline recognizer, rather than by hand transcription. The hypothesis that minimizes this equation, y, is considered the gold standard for perceptron and GCLM training. This allows the use of more training data for which hand transcriptions are not available, and as one expects, the additional training data improves WER scores.

Arisoy et al. [4] revisits discriminative language modeling with a Turkish ASR application. The rich morphology of Turkish serves as a good example of how discriminative LMs, in particular those that include morphological as well as whole-word features, might work with similarly rich Semitic languages. In this study, the authors experiment with features derived from word n-grams, word-based syntactic features, morph n-grams (derived via Morfessor [17]), syntactic features estimated over the morph sequences, and topic-based features estimated using a *tf-idf* vector methodology. The results show that while morph-based features and syntactic features (both word- and morph-based) in the discriminative LM framework improve ASR results, the semantically-informed topic features and long-distance morph relation features are not useful. It is also the case that unigram features are better than higher-order n-gram features in the discriminative LM. The discriminative modeling and morph-based features help overcome the OOV problem and data sparsity that occur in Turkish, and the lessons may well be applied to building language models for Arabic and other Semitic languages for ASR or other NLP applications.

5.5.10 LSA Language Models

Bellegarda [7] describes a method for integrating long-range semantic information into the normally short-range context of n-gram models. This method first involves calculating the Singular Value Decomposition (SVD) of a word-by-document matrix derived from a large set of training data that is semantically similar to the test data. The decomposed and truncated matrices derived via SVD can be used to represent the words and documents in a clustered semantic space. In other words, it becomes possible to compare each word to other words in the training set based on their distribution among documents. The decomposition effectively takes into account all word pairs and all document pairs in deriving the similarity of all words and documents. A new document can be clustered into a semantic grouping of similar documents from the training set by projecting it into the SVD space. In terms of n-gram modeling, we can model the probability of each word in a document given the semantic history of the entire preceding document. This semantic language model prediction is interpolated with typical n-gram modeling in the following way:

$$P(w_i|H_{i-1}^{(n+l)}) = \frac{p(w_i|w_{i-n+1}^{i-1})\frac{p(w_i|\tilde{d}_{i-1})}{p(w_i)}}{\sum_{w_i \epsilon V} p(w_i|w_{i-n+1}^{i-1})\frac{p(w_i|\tilde{d}_{i-1})}{p(w_i)}} \tag{5.23}$$

where w_i is the current word, $p(w_i|\tilde{d}_{i-1})$ is the LSA model probability for the word given the semantic history, and V is the vocabulary. $H_{i-1}^{(n+l)}$ represents the integration of the n-gram probability n with an LSA probability l for the history of the document up until word w_i. This calculation provides a natural interpolation of long- and short-term context. The component $\tilde{d}_{i-1}$ can be smoothed by referring to the word or document clusters rather than to the vectors of the individual words seen in the document. Further tuning can be applied to determine how much of the previous context should be taken into account for each word.

The experiments in [7] show good reductions in perplexity and word error rate when incorporating the LSA history into the standard bigram and trigram model. However, some caveats apply. The semantic type of the training data used to develop the LSA model must be tightly coupled with that of the test data, otherwise the effectiveness of the extra modeling is greatly diminished. Luckily, not a very large amount of data is necessary to construct an effective model. Also, while the LSA modeling is very useful as concerns content words, those that control the broad semantic topics and ideas of a document, it is not effective at all in discriminating between function words that appear in all documents. As these tend to be the words most likely to be mis-recognized in an ASR context, the benefit of LSA may be limited for that task. However, while function word mistakes may be common, mistakes in content words may be more harmful to actual understanding, so that the mistakes that are avoided by using an LSA-interpolated model may outweigh the function word mistakes in some evaluations. Furthermore, the long-distance semantic system can also be interpolated with a near-term, syntactically trained language model.

The Semitic NLP literature is missing reference to LSA language modeling. This may be because a major obstacle in Semitic NLP is often the precise word choice, i.e. choosing the right syntactic word form from among many semantically identical ones, for which LSA modeling is not helpful. However, in natural language generation, LSA modeling may help to provide some variation in phrase creation, and it could also be useful in English-to-Semitic machine translation.

5.5.11 Bayesian Language Models

Two related goals motivate the language modeling work of [66]: to incorporate a corpus-based topic model into an n-gram language model, and to do so using a Bayesian algorithm. The large-context topic modeling serves to complement the smaller-context n-gram modeling. The Bayesian paradigm enforces a method in which all assumptions about the data are formulated mathematically in order to be incorporated into the model. The work of [66] is based largely on that of [40]; the discussion below draws from both studies.

The path to achieving both goals is found in Dirichlet modeling, which is used as a replacement for the simpler kinds of smoothing and backoff described in Sect. 5.4. A Dirichlet model provides a prior model of the data. Assume a matrix Φ that describes word transition probabilities over a training corpus. Each row w' of Φ represents the probabilities $P(w|w')$. These rows make up the parameters of the model Q. However, the parameters of Q are usually poorly estimated because of the sparse data problem that affects all language modeling attempts. The real parameters of the data are unknown; we can easily imagine that there are multiple possible parameters of Q. The *distribution* of the possible parameters of Q is estimated by the Dirichlet model. The parameters of the Dirichlet model – a hierarchical, exponential model – are the *hyperparameters* of the corpus data. The crucial parameters of the Dirichlet model are usually notated as α and m; α describes the peakiness of the distribution, while m describes its mean. In most of Chap. 3 of her dissertation, Wallach replaces these with β and n, respectively, and that is the notation that will be used in this summary. The focus here is the *use* of the Dirichlet model in language and topic modeling; for a further explanation of its properties and estimation, please see [66] and [40].

Typical bigram modeling starts with the same transition matrix Φ, using a formulation like Eq. (5.24) to predict the probability of a test corpus given the training data:

$$P(\mathbf{w}|\Phi) = \prod_{w} \prod_{w'} \phi_{w|w'}^{N_{w|w'}} \tag{5.24}$$

The term $\phi_{w|w'}$ describes the probability of the bigram $w'w$ in the training data, while the term $N_{w|w'}$ is an exponent of ϕ describing the bigram's count in test data. Smoothing and backoff algorithms such as those described in Sect. 5.4 are

usually used to enhance this formulation to account for the sparse data problem. Those methods are replaced here with a Dirichlet prior, which counters the sparse data problem by estimating the probability of our training data Φ over the hyperparameters βn of the Dirichlet model:

$$P(\Phi|\beta n) = \prod_{w'} Dir(\phi_{w'}|\beta n) \tag{5.25}$$

When we substitute Eq. (5.25) into Eq. (5.24), we can estimate $P(\mathbf{w}|\beta n)$, the probability of the test corpus given the Dirichlet hyperparameters:

$$P(\mathbf{w}|\beta n) = \prod_{w'} \frac{\prod_w \Gamma(N_{w|w'} + \beta n_w)}{\Gamma(N_{w'} + \beta)} \cdot \frac{\Gamma(\beta)}{\prod_w \Gamma(\beta n_w)} \tag{5.26}$$

Γ is a function used in the estimation of the Dirichlet model. At the test stage, we can predict the probability of word w given a previous word w' and the estimated model using the following formulation:

$$P(w|w', \mathbf{w}, \beta n) = \frac{N_{w|w'} + \beta n_w}{N_{w'} + \beta} \tag{5.27}$$

$$= \lambda_{w'} n_w + (1 - \lambda_{w'}) \frac{N_{w|w'}}{N_{w'}} \tag{5.28}$$

Equation (5.28) is meant to evoke a typical interpolation model: $\frac{N_{w|w'}}{N_{w'}}$ is a bigram probability; we back off to a unigram probability described by n_w, and these are interpolated by parameter $\lambda_{w'}$. In the Dirichlet formulation, the concentration (peakiness) hyperparameter β takes the place of EM-estimated λ, and the n_w hyperparameter takes the place of Good-Turing or another kind of smoothing discounting for w. The result is a formulation of the smoothing parameters of the bigram $w'w$ through the calculation of a Dirichlet prior rather than through EM estimation.

Similar intuitions are used to apply the Dirichlet prior to topic modeling. This is known as latent Dirichlet allocation, and is also described in [9]. Earlier we described a matrix Φ in which the rows represent the probabilities of transition from word w' to each word w. Now we instead define Φ as the probabilities of transitioning from topic t to words w. The set of topic distributions are notated as $\mathbf{z}$. Furthermore, we assume that a given document will include a subset of possible topics, so there is a second transition matrix Θ from documents to topics. The estimation of each of these two matrices is again based on sparse data, and therefore it is useful to estimate a Dirichlet prior describing their distributions. We estimate hyperparameters β and n to describe the distribution of possible matrices Φ, and separate hyperparameters α and m to describe the distribution of possible matrices Θ.

To estimate the probability of a word using a topic-based model, we apply the parameters β and n:

$$P(w|t, \mathbf{w}, \mathbf{z}, \beta n) = \frac{N_{w|t} + \beta n_w}{N_{*|t} + \beta} \tag{5.29}$$

$$= \lambda_t n_w + (1 - \lambda_t) \frac{N_{w|t}}{N_t} \tag{5.30}$$

The term $N_{*|t}$ refers to the distribution of all words over that topic t. Similarly, to calculate the probability of a given topic given a document model, we apply the hyperparameters α and m:

$$P(t|d, \mathbf{w}, \mathbf{z}, \alpha m) = \frac{N_{t|d} + \alpha m_t}{N_{*|d} + \alpha} \tag{5.31}$$

$$= \lambda_d m_t + (1 - \lambda_d) \frac{N_{t|d}}{N_d} \tag{5.32}$$

The models resulting from latent Dirichlet allocation have properties in common with those resulting from latent semantic analysis, described in Sect. 5.5.10.

Given a topic model and a bigram model, the next challenge is to find a method to combine their predictions. When using LSA to incorporate topical information (Sect. 5.5.10), the topic models are combined with n-gram models using linear interpolation. In contrast, to retain and extend the Bayesian properties of the component n-gram and topic models described here, [66] calculates the joint probability of a word given both its n-gram context and its topic context by combining Eqs. (5.27) and (5.29). Stepping back to the initial model of the training corpus, the transition probability matrix Φ will now have rows defined as the probability of transition from word w' *and* topic t to word w: $P(w|w't)$. The Dirichlet hyperparameters describe the probability distribution of this jointly estimated matrix Φ. If the contexts $w't$ are modeled as strictly joint, then the estimation is formulated as:

$$P(\Phi|\beta n) = \prod_{w'} \prod_{t} Dir(\phi_{w't}|\beta n) \tag{5.33}$$

The hyperparameters β and n are fully tied, shared between the contexts w' and t. Alternatively, the contexts w' might be modeled separately for each topic:

$$P(\Phi|\{\beta n_t\}_{t=1}^{T}) = \prod_{w'} \prod_{t} Dir(\phi_{w't}|\beta n_t) \tag{5.34}$$

Here the hyperparameters capture topic-specific similarities between words. Lastly, the distributions of t could be modeled separately over each context w':

$$P(\Phi|\{\beta n_{w'}\}_{w'=1}^{W}) = \prod_{w'} \prod_{t} Dir(\phi_{w't}|\beta n_{w'}) \tag{5.35}$$

In this last case, the parameters capture information about common, topic-independent bigrams.

The choice of prior as well as the choice of estimation procedure affect the outcome, as better choices result in more interpretable topics and more informative models. For jointly estimating word and topic contexts, [66] shows Eq. (5.34) to be most effective; it is more effective than Dirichlet-estimated bigram or topic models alone, and more effective than using either of the other priors (Eqs. 5.33 and 5.35) to estimate the joint model.

The models are evaluated by a measure related to perplexity, differing from the usual formulation due to the use of topic models and the alternative method of estimating the models. The results over a small training and test set show that the proposed algorithms are effective in reducing the information rate of the models. Furthermore, the words comprising topic models are interpretable, showing that the method is as useful as LSA in incorporating semantic properties of the corpus into the language model. The differences in evaluation technique, however, preclude the comparison of the Bayesian topic-based models in [66] to typical language modeling methods. Ostensibly, this method of calculating the models and of testing them is task-independent, but no task-based evaluation is given. The questions of whether this technique is amenable to use in speech recognition or machine translation in terms of storage and processing requirements, and whether the information theoretic gains extend to gains in task results, remain for further research.

As for tasks within the Semitic language realm, the usual questions apply: should the word contexts be determined over unique tokens, or should stemming be applied first? More generally, will using this technique for estimating smoothing parameters be useful in solving the sparse data problem that most affects Semitic languages – the growth of vocabulary due to morphological richness? Perhaps the incorporation of a third kind of context, one that models morphological properties, will result in the most useful model for Semitic language applications.

5.6 Modeling Semitic Languages

The specifics of morphological and grammatical principles of Semitic languages are covered elsewhere in this volume. There are multiple morphological functions, including both root-and-pattern and affixational morphology, that result in an abundance of unique word forms. As for handling this rich morphology in automatic processing, there are two overall devices: tokenization, which reduces the number of unique words by separating clitics from the main semantic information, and stemming, which reduces the number of unique word forms by arranging words into classes by their common properties, ignoring the less semantically salient properties that make them different. Tokenization can often be applied through simple scripts that look for specific alphabetic sequences at word boundaries. An information-theoretic approach to tokenization can also be taken when affix and stem dictionaries are unavailable (e.g. Morfessor, [17]). There are multiple

approaches to stemming, ranging from language-independent statistical methods of deriving morphemes to linguistically complex methods such as the multi-featured, context-sensitive classes derived through the use of parsers and part-of-speech taggers for Factored Language Models [33]. This section will review studies that incorporate tokenization, stemming, or both in order to improve language models for Semitic natural language processing.

5.6.1 Arabic

Vergyri et al. [65] and Kirchhoff et al. [33] both describe the utility of incorporating morpheme information into language models for Arabic speech recognition. Both of these studies use Egyptian Colloquial Arabic data and Factored Language Models (FLMs), described above in Sect. 5.5.4. Automatic means are used to derive morphemes as well as other kinds of morphological and syntactic information. These studies focus on how FLMs and associated technologies such as generalized parallel backoff can be used to successfully incorporate this knowledge into a useful language model, eventually reducing word error rate on an ASR task. One obstacle that must be overcome is the combination of recognized morphemes into words without generating spurious words, or words not in the original vocabulary. Another aspect of this technology is the decision of when to incorporate the morpheme models; if the model is too complex to be effectively incorporated into first-pass decoding, it can still be beneficial to use morpheme and enhanced models in a second-pass decoding of word-based lattices. The experiments in [33] also effectively separate the aspects of morpheme-based language modeling from the FLM technique in a useful way; it is clear in this study that the use of the more complex FLMs indeed improves upon a more simple incorporation of morphological knowledge into the language model.

The use of morphemes in Arabic language modeling is also discussed in [21], discussed in Sect. 5.5.5 above. Morphemes derived via a finite state algorithm, part of speech tags, and automatically-derived short vowel information are used as input to a neural network language modeling tool. While the gain in word error rate is negligible in this study, the follow-up study [38] does show that accurately derived morphological information can be used to successfully lower word error rate when incorporated into neural network language models.

When working with a dialectal form of Arabic, there is a tension between using morphological information derived from the more data- and resource-replete Modern Standard Arabic and creating new tools to work with the particular dialect, for which there may not be adequate text for training new models. Afify et al. [2] combine tools and knowledge from both MSA and Iraqi Arabic in creating morpheme-based language models for the Iraqi dialect. First, a set of affixes are predefined based on linguistic knowledge of the dialect. The affixes are segmented from the words based on orthographic information. These segmentations are then refined by applying knowledge from the Buckwalter Arabic Morphological Analyzer [11],

which encodes information about legal stems in MSA. Heuristics such as setting a minimum stem length and checking stem accuracy against the stem dictionary limit the number of segmentations of each word. The resulting segmentations are used to derive both the language model and pronunciation dictionaries. The morpheme models result in a lower out-of-vocabulary rate and a lower ASR word error rate, especially when interpolation is used to smooth the word and morpheme language models. The interpolation allows the model to take advantage of the longer context of word models and the greater token coverage of morpheme models.

Xiang et al. [68] use a similar method of segmentation constrained by defined MSA stems to produce morpheme-based language models of Arabic. When automatically derived phonetic information is incorporated into both these language models and corresponding acoustic models, word error rates drop. An important observation of this study is that words must be segmented carefully; doing so without constraint results in models with greater acoustic confusability. Incorporating phonetic knowledge into the segmentation algorithm aids in producing models that are better aligned with the challenges encountered in ASR. Constrained segmentation of Arabic words into morphemes is also explored in [39] and [16]. In these studies, the statistically-derived morpheme segmentations are constrained by pre-defined affix and stem lists. The benefit of using these morphemes in language modeling for ASR may be limited to small-vocabulary tasks [16], but in some cases may also be useful in large-vocabulary tasks [46].

Heintz [27] uses finite state machines to identify all of the possible stems in a word according to a set of pattern templates. Together with a set of predefined affixes, the segmentation into morphemes produces a text over which useful language models can be estimated. However, there is no significant reduction in word error rate in either Modern Standard Arabic or Levantine Arabic speech recognition experiments. The use of simpler techniques such as Morfessor [17] or affix-splitting for morpheme creation is recommended.

Marton et al. [42] discusses the use of tokenization of an Arabic language model for translation from English to Arabic. (The main thesis of this study regards paraphrasing the source language, English, but this does not affect the language modeling of the target language, Arabic.) Tokenization and normalization are performed using MADA, a morphological analysis tool trained to work with Modern Standard Arabic data. They note that the process of *de-tokenization*, returning the morphemes to word status by correctly strining them back together, is a non-trivial and important step in processing, and discuss a successful technique.

5.6.2 Amharic

In [1], bigram language models are incorporated into an Amharic large-vocabulary automatic speech recognition system. The system is straightforward and without complications of morphological processing. The use of only whole words results in a rather weak model due to data sparsity – both perplexity and the number of

singleton words are large. The author points to the use of syllable modeling as a useful prospect for segmenting Amharic text; the use of syllables in the acoustic modeling experiments, tested separately from language modeling, is shown to be beneficial.

Tachbelie and Menzel [63] build on the work for Amharic NLP begun in [1]. In this study, the authors use the Morfessor algorithm [17] to segment the Amharic train and test texts, greatly reducing the out-of-vocabulary rate. A comparison of word and morpheme language models via log probability (see Sect. 5.2) shows that the word-based model provides a better encoding of the test text. Still, reducing the out-of-vocabulary rate should be useful for application of the model in an NLP task, and the language-independent Morfessor algorithm is sufficient to provide adequate segmentation for this purpose. The authors test their hypotheses further in [64]. Here the speech recognition system developed in [1] is used to test several morpheme-based language models in Amharic. A manual segmentation of words into morphemes is compared to the Morfessor-derived morphs. The morpheme models are applied by expanding word bigram lattices into morpheme-based lattices and finding the new best path. Linguistically (manually) derived morphemes, and trigrams in particular, are found to produce better results in this second-pass decoding. Factored language models are also explored, with word, POS, prefix, root, pattern and suffix features. Pre-determined and genetic algorithm backoff models are applied in this study. FLMs, especially those that include POS information and use genetic algorithms to find the best backoff path, produce better decoding results than the simpler methods described above.

Pellegrini and Lamel [48] also study the application of NLP techniques to Amharic. Words are decomposed using the Harris algorithm, which looks for morpheme boundaries at places where the number of possible subsequent letters is large. Splitting words at these affix boundaries effectively reduces the size of the vocabulary. Furthermore, ASR experiments using re-trained morpheme acoustic models are successful in reducing the word error rate, both when morphemes are counted as ‘words’ and when the morphemes are recombined into words. However, it is found in later studies that this result is restricted to this experiment with its small 2-h acoustic model training set. In [49], the word decompounding is performed with Morfessor, modified to include a Harris probability for word boundaries. The effect of this is to favor short morphs, which correspond better to the kind of concatenative morphemes (affixes) that are found in Semitic languages. Furthermore, the algorithm is enhanced with information about phonetic distinctive features. A constraint is added to each model regarding acoustic confusability, derived from acoustic alignment tables. This biases the algorithm to avoid creating monosyllables that are acoustically confusable. All of the morpheme models reduce the vocabulary size of the training set as compared to word models. In applying these models in an ASR task, the lowest word error rate occurs with the most complex model, which uses both morphological and phonetic constraints in its segmentation. This study uses a 35-h training set, and finds that the use of a phonetic constraint is required when deriving morphemes for language modeling to reduce word error rate in Amharic ASR.

5.6.3 Hebrew

Netzer et al. [43] use a trigram-model trained on 27 million Hebrew words for the task of predicting likely words following a given context. The task envisioned in this study is mobile device input, therefore no acoustic or translation models are necessary. A morphological and syntactic model are incorporated to account for rich Hebrew morphology, but in this case, results are worsened. The application is such that only a partial sentence is used in computing and applying morphological and syntactic information at test time, which likely accounts for the unexpected results. It is important to have sufficient context when applying morphological and syntactic models to modify or enhance both training and test texts.

Shilon et al. [58] is a study of Hebrew-Arabic machine translation. There is great difficulty in building such a system, as there is no sufficient corpus of parallel language data. Target-language LMs can be built on the available single-corpus data, but it is also essential to incorporate linguistic rules in the decoding phase that account for the morphological and syntactic differences that exist between Hebrew and Arabic. A second difficulty is the fact that the lattices output by the decoder tend to be very large, both for Hebrew and Arabic as the target language. Again, this is due to the morphological richness of both languages, and the lack of parallel corpora to help align and filter the output. In general, the language models are not able to sufficiently assign accurate probabilities to all of the sequences in the lattices. This study points to the amount of work left to be realized, both in building and applying resources like parallel corpora and knowledge-rich language models, for the successful application of statistical machine translation in Arabic and Hebrew.

5.6.4 Maltese

The morphological characteristics of Maltese are somewhat different than those of the other Semitic languages discussed in this book. This impacts the kinds of techniques that might be used in language modeling for Maltese.

As stated earlier in the book, Maltese is written with Latin text. Unlike Modern Standard Arabic or Hebrew, all vowels are included in the orthographic form. If vowel variation were to occur within stems as it does in the derivational morphology of Hebrew and Arabic, this would produce a great increase in the number of word forms in Maltese. However, it is the case that vowel harmony is not used for morphological purposes, so vocabulary size should not increase dramatically due to the inclusion of vowels in the orthography.

The problem of vocabulary expansion in Modern Standard Arabic and other Semitic languages comes from the combinations of many affixes and clitics used to encode morphological information such as gender and number agreement, case markings, object pronouns, tense, and aspect. Maltese also uses affixes for some of these purposes, so that the use of a stemmer may reduce the out-of-vocabulary rate.

However, Maltese uses affixational morphology somewhat less frequently than is seen in other Semitic languages. For instance, there is no case marking on nouns, and some tense and aspect markers are separate particles, meaning that the amount of vocabulary expansion will not be as great for Maltese. Therefore, larger training sets may compensate for the growth in vocabulary due to affixational morphology more quickly than is the case in MSA. If adequate training data is not available, stemming techniques are likely to be useful in generating the most reliable language models.

It is also the case that Maltese has relatively free word order. The ordering of verb, subject, and object changes based on the topic of the sentence. This free word order means that exact n-grams are less likely to be repeated throughout a corpus. A modeling technique like skip n-grams or flexgrams (Sect. 5.5.1) may be especially useful in Maltese for capturing the history of a word without relying too heavily on word order.

5.6.5 Syriac

It is likely that the most pressing issue in language modeling for Syriac will be the lack of training data. This issue will be exacerbated by the existence of several forms of orthography. Therefore, a crucial step will be the normalization of the orthography in order to take advantage of all available data.

As in MSA and Hebrew, many vowels are not normally marked in the orthography, reducing the number of word forms that must be predicted. Affixational morphology is used on verbs in Syriac to encode number, gender, tense, and object pronouns, as well as possessive pronouns on nouns. Given the probable lack of training data, stemming will be useful to decrease the out-of-vocabulary rate. The regularity of the affixes and clitics may mean that a surface segmenter such as Morfessor [17] will be adequate to provide useful morphemes, without resorting to a deep morphological analysis.

5.6.6 Other Morphologically Rich Languages

Languages such as Turkish and Finnish also exhibit high degrees of morphological complexity resulting in a quickly expanding vocabulary and high out-of-vocabulary rate [25, 28]. Many of the techniques for dealing with the resulting data sparsity in Semitic languages can be used to address the same problem in these other languages, and vice versa. For instance, Sect. 5.5.9 mentions the work of [4] in which morpheme and lexical information is used in a language model to improve the results of a Turkish ASR task. Hacioglu et al. [26] also describe a morpheme-splitting algorithm for Turkish, where the morphemes rather than words are again used to improve ASR. Siivola et al. [59] discuss the use of stemming in building

language models for Finnish ASR. Hirsimäki et al. [28] introduced the Morfessor algorithm for segmenting words into statistically-derived "morphs" by applying it to Finnish data. These and other studies on morphologically rich languages can certainly inform natural language processing work in Semitic languages.

5.7 Summary

In this chapter we have explored basic n-gram modeling and many variations on that technique that allow the incorporation of semantic, syntactic, and even phonetic knowledge into the models. Within Semitic language modeling, the focus has been on how best to acquire and make use of morphological information to provide context and knowledge for the language models. Modeling techniques that make this incorporation simpler, in particular factored language modeling and neural network modeling, seem most promising for Semitic applications.

The use of one model over the other will often depend not only on the task, but also on the resources available. When working with languages for which resources are scarce, simpler linguistic extraction methods, such as statistical segmenting, often do a sufficiently accurate job to provide informative features to the language models. Exploration of the simpler expanded language modeling techniques, such as skip n-grams and flex-grams, may be of greatest use in languages like Amharic, Syriac, and Maltese, where corpus data and automatic tools for parsing, etc. are difficult to acquire. These techniques have been shown to be as effective in improving language models as adding more data – a crucial factor when very little data is available.

When corpus data as well as automatic means for extracting part-of-speech labels, roots, or other syntactic or morphological information are available, then the more complex modeling methods that can incorporate this data will have an advantage. Further study of more complex methods such as Bayesian topic-based language modeling may be appropriate for languages such as Modern Standard Arabic and Modern Hebrew, for which data and tools are available.

References

1. Abate ST (2006) Automatic speech recognition for Amharic. PhD thesis, Universität Hamburg
2. Afify M, Sarikaya R, Kuo HKJ, Besacier L, Gao Y (2006) On the use of morphological analysis for dialectal Arabic speech recognition. In: Proceedings of ICSLP, Pittsburgh
3. Al-Haj H, Lavie A (2012) The impact of Arabic morphological segmentation on broad-coverage English-to-Arabic statistical machine translation. Mach Trans 26:3–24
4. Arisoy E, Saraçlar M, Roark B, Shafran I (2012) Discriminative language modeling with linguistic and statistically derived features. IEEE Trans Audio Speech Lang Process 20(2):540–550

5. Bahl LR, Brown PF, Souza PVD, Mercer RL (1989) A tree-based statistical language model for natural language speech recognition. IEEE Trans Acoust Speech Signal Process 37:1001–1008
6. Bell T, Cleary J, Witten I (1990) Text compression. Prentice Hall, Englewood Cliffs
7. Bellegarda J (2000) Exploiting latent semantic information in statistical language modeling. Proc IEEE 88(8):1279–1296
8. Berger AL, Pietra SAD, Pietra VJD (1996) A maximum entropy approach to natural language processing. Comput Linguist 22(1):39–71
9. Blei DM, Ng AY, Jordan MI (2003) Latent Dirichlet allocation. J Mach Learn Res 3:993–1022
10. Brown P, Pietra VD, deSouza P, Lai J, Mercer R (1992) Class-based *n*-gram models of natural language. Comput Linguist 18:367–379
11. Buckwalter T (2004) Buckwalter Arabic morphological analyzer version 2.0. http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2004L02
12. Chelba C, Jelinek F (2000) Structured language modeling. Comput Speech Lang 14:283–332
13. Chelba C, Brants T, Neveitt W, Xu P (2010) Study on interaction between entropy pruning and Kneser-Ney smoothing. In: INTERSPEECH 2010, Makuhari, Chiba, pp 2422–2425
14. Chen S, Goodman J (1999) An empirical study of smoothing techniques for language modeling. Comput Speech Lang 13:359–394
15. Chen S, Mangu L, Ramabhadran B, Sarikaya R, Sethy A (2009) Scaling shrinkage-based language models. In: Automatic speech recognition & understanding, Merano/Meran, pp 299–304
16. Choueiter G, Povey D, Chen SF, Zweig G (2006) Morpheme-based language modeling for Arabic LVCSR. In: Proceedings of ICASSP, Toulouse, pp 1053–1056
17. Creutz M, Lagus K (2007) Unsupervised models for morpheme segmentation and morphology learning. ACM Trans Speech Lang Process 4:1–34
18. Deligne S, Bimbot F (1995) Language modeling by variable length sequences: theoretical formulation and evaluation of multigrams. In: Proceedings of ICASSP, Detroit, vol 1, pp 169–172
19. Duh K, Kirchhoff K (2004) Automatic learning of language model structure. In: Proceedings of COLING 2004, Geneva
20. El Kholy A, Habash N (2012) Orthographic and morphological processing for English-Arabic statistical machine translation. Mach Trans 26:25–45
21. Emami A, Zitouni I, Mangu L (2008) Rich morphology based n-gram language models for Arabic. In: Interspeech 2008, Brisbane, pp 829–832
22. Gale WA, Church KW (1994) What's wrong with adding one? In: Oostdijk N, De Haan P (eds) Corpus-based resarch into language. Rodolpi, Amsterdam
23. Gale WA, Sampson G (1995) Good-turing frequency estimation without tears. J Quant Linguist 22:217–37
24. Guthrie D, Allison B, Liu W, Guthrie L, Wilks Y (2006) A closer look at skip-gram modelling. In: Proceedings of LREC, Genoa
25. Guz U, Favre B, Hakkani-Tür D, Tur G (2009) Generative and discriminative methods using morphological information for sentence segmentation of Turkish. IEEE Trans Audio Speech Lang Process 17(5):895–903
26. Hacioglu K, Pellom B, Ciloglu T, Ozturk O, Kurimo M, Creutz M (2003) On lexicon creation for Turkish LVCSR. In: Proceedings of Eurospeech '03, Geneva, pp 1165–1168
27. Heintz I (2010) Arabic language modeling with stem-derived morphemes for automatic speech recognition. PhD thesis, The Ohio State University
28. Hirsimäki T, Creutz M, Siivola V, Kurimo M, Viripioja S, Pylkkönen J (2006) Unlimited vocabulary speech recognition with morph language models applied to Finnish. Comput Speech Lang 20:515–541
29. Jelinek F, Mercer R (1980) Interpolated estimation of Markov source parameters from sparse data. In: Proceedings of the workshop on pattern recognition in practice, Amsterdam, pp 381–397
30. Katz SM (1987) Estimation of probabilities from sparse data for the language model component of a speech recognizer. IEEE Trans Acoust Speech Signal Process ASSP-35(3):400–401

31. Khudanpur S, Wu J (2000) Maximum entropy techniques for exploiting syntactic, semantic, and collocational dependencies in language modeling. Comput Speech Lang 14:355–372
32. Kirchhoff K, Bilmes J, Henderson J, Schwartz R, Noamany M, Schone P, Ji G, Das S, Egan M, He F, Vergyri D, Liu D, Duta N (2002) Novel approaches to Arabic speech recognition: report from the 2002 Johns-Hopkins summer workshop. Technical report, Johns Hopkins University
33. Kirchhoff K, Vergyri D, Bilmes J, Duh K, Stolcke A (2006) Morphology-based language modeling for conversational Arabic speech recogntion. Comput Speech Lang 20:589–608
34. Kneser R (1996) Statistical language modeling using a variable context length. In: Proceedings of the fourth international conference on speech and language processing, Philadelphia, pp 494–497
35. Kneser R, Ney H (1995) Improved backing-off for m-gram language modeling. In: Proceedings of the IEEE international conference on acoustics, speech and signal processing, Detroit, vol 1, pp 181–184
36. Koehn P (2010) Statistical machine translation. Cambridge University Press, Cambridge
37. Kuo HK, Arisoy E, Mangu L, Saon G (2011) Minimum Bayes risk discriminative language models for Arabic speech recognition. In: IEEE workshop on ASRU, Waikoloa, pp 208–213
38. Kuo HKJ, Mangu L, Emami A, Zitouni I (2010) Morphological and syntactic features for Arabic speech recognition. In: Proceedings of ICASSP, Dallas, pp 5190–5193
39. Lee YS, Papineni K, Roukos S, Emam O, Hassan H (2003) Language model based Arabic word segmentation. In: Proceedings of the 41st annual meeting for ACL, Sapporo, vol 1, pp 399–406
40. Mackay DJC, Bauman Peto LC (1995) A hierarchical Dirichlet language model. Nat Lang Eng 1(3):289–307
41. Malouf R (2002) A comparison of algorithms for maximum entropy parameter estimation. In: Proceedings of Co-NLL, Taipei
42. Marton Y, el Kholy A, Habash N (2011) Filtering antonymous, trend-contrasting, and polarity-dissimilar distributional paraphrases for improving statistical machine translation. Proceedings of 6th workshop on statistical machine translation, Edinburgh, pp 237–249
43. Netzer Y, Adler M, Elhadad M (2008) Word prediction in Hebrew – preliminary and surprising results. In: ISAAC 2008, Gold Coast. http://www.cs.bgu.ac.il/~adlerm
44. Ney H, Essen U (1991) On smoothing techniques for bigram-based natural language modelling. In: Proceedings of the IEEE international conference on acoustics, speech and signal processing '91, Toronto, vol 2, pp 825–829
45. Ney H, Essen U, Kneser R (1994) On structuring probabiliistic dependences in stochastic language modelling. Comput Speech Lang 8:1–38
46. Nguyen L, Ng T, Nguyen K, Zbib R, Makhoul J (2009) Lexical and phonetic modeling for Arabic automatic speech recognition. In: INTERSPEECH 2009, Brighton, pp 712–715
47. Niesler TR, Woodland P (1996) A variable-length category-based n-gram language model. In: Proceedings of ICASSP, Atlanta
48. Pellegrini T, Lamel L (2006) Investigating automatic decomposition for ASR in less represented languages. In: INTERSPEECH-2006, Pittsburgh, pp 1776–1779
49. Pellegrini T, Lamel L (2007) Using phonetic features in unsupervised word decompounding for ASR with application to a less-represented language. In: INTERSPEECH-2007, Antwerp, pp 1797–1800
50. Roark B, Saraçlar M, Collins M (2007) Discriminative *n*-gram language modeling. Comput Speech Lang 21:373–392
51. Ron D, Singer Y, Tishby N (1996) The power of amnesia: learning probabilistic automata with variable memory length. Mach Learn 25:117–149
52. Rosenfeld R (1996) A maximum entropy approach to adaptive statistical language modelling. Comput Speech Lang 10:187–228
53. Sarikaya R, Deng Y (2007) Joint morphological-lexical language modeling for machine translation. In: NAACL-Short '07 human language technologies 2007, Rochester, pp 145–148
54. Sarikaya R, Afify M, Gao Y (2007) Joint morphological-lexical language modeling (JMLLM) for Arabic. In: Proceedings of ICASSP, Honolulu, pp 181–184
55. Schwenk H (2007) Continuous space language models. Comput Speech Lang 21(3):492–518

56. Sethy A, Chen S, Ramabhadran B (2011) Distributed training of large scale exponential language models. In: Proceedings of ICASSP, Prague, pp 5520–5523
57. Seymore K, Rosenfeld R (1996) Scalable backoff language models. In: Proceedings of ICSLP 1996, Philadelphia, pp 232–235
58. Shilon R, Habash N, Lavie A, Wintner S (2012) Machine translation between Hebrew and Arabic. Mach Trans 26:177–195
59. Siivola V, Kurimo M, Lagus K (2001) Large vocabulary statistical language modeling for continuous speech recognition in Finnish. In: Proceedings of the 7th European conference on speech communication and technology, Copenhagen, pp 737–747
60. Siivola V, Creutz M, Kurimo M (2007) Morfessor and VariKN machine learning tools for speech and language technology. In: INTERSPEECH 2007, Antwerp, pp 1549–1552
61. Siivola V, Hirsimäki T, Virpioja S (2007) On growing and pruning Kneser-Ney smoothed N-gram models. IEEE Trans Audio Speech Lang Process 15(5):1617–1624
62. Stolcke A (1998) Entropy-based pruning of backoff language models. In: Proceedings of the DARPA broadcast news transcription and understanding workshop, Virginia, pp 270–274
63. Tachbelie MY, Menzel W (2007) Sub-word based language modeling for Amharic. In: Proceedings of international conference on recent advances in NLP, Borovets, pp 564–571
64. Tachbelie MY, Abate ST, Menzel W (2009) Morpheme-based language modeling for Amharic speech recognition. In: The 4th language and technology conference, Poznan
65. Vergyri D, Kirchhoff K, Duh K, Stolcke A (2004) Morphology-based language modeling for Arabic speech recognition. In: Proceedings of ICSLP, Jeju Island, pp 2245–2248
66. Wallach HM (2008) Structured topic models for language. PhD thesis, University of Cambridge
67. Witten I, Bell T (1991) The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression. IEEE Trans Inf Theory 37:1085–1094
68. Xiang B, Nguyen K, Nguyen L, Schwartz R, Makhoul J (2006) Morphological decomposition for Arabic broadcast news transcription. In: Proceedings of ICASSP, Toulouse, pp 1089–1092
69. Yuret D, Biçici E (2009) Modeling morphologically rich languages using split words and unstructured dependencies. In: Proceedings of the ACL-IJCNLP 2009 conference short papers, Singapore, pp 345–348
70. Zitouni I, Zhou Q (2007) Linearly interpolated hierarchical n-gram language models for speech recognition engines. In: Robust speech recognition and understanding. I-Tech, Vienna, pp 301–318