16

Language Modelling for Multilingual Speech Translation

Fuliang Weng Andreas Stolcke
SRI International, Menlo Park

Michael Cohen
Nuance Communications, Menlo Park

16.1 Introduction

As we saw in Chapter 14, the speech recognition problem can be formulated as the search for the best hypothesised word sequence given an input feature sequence. The search is based on probabilistic models trained on many utterances:

$$W = \underset{W}{argmax}\, P(X \mid W)P(W)$$

In the equation above, $P(X \mid W)$ is called the acoustic model, and $P(W)$ is called the language model (LM).

In this chapter we present several techniques that were used to develop language models for the speech recognisers in the SLT system. The algorithms presented here deal with two main issues: the data-sparseness problem and the development of language models for multilingual recognisers.

As with acoustic modelling, sparse training data is one of the main problems in language modelling tasks. In both cases, we ideally want to have enough properly matched data to train models for all the necessary conditions. One may think that today's technology, especially the Internet and the World Wide Web, lets us take for granted the availability of any amount of language modelling training data. Unfortunately, this is not entirely true, for three reasons:

Style mismatch: Internet-derived data is usually written text, which does not have the same style as spoken material.

Language mismatch: The available texts are not uniformly distributed with respect to different languages: there is plenty of data available for English, but not for other languages.

Domain mismatch: The texts are not specifically organized for any speech recognition task.

Ignoring these mismatches can cause significant degradation to the performance of speech recognition systems. On the other hand, fully satisfying them may introduce data-sparseness problems.

For these reasons, we want to investigate various ways that allow us to rapidly collect proper amounts of data for new domains and new languages, and to train compact language models with high performance. Since human beings are the ultimate producers and consumers of human languages, it is also desirable to leverage the strength of symbolic approaches for statistical language modelling.

We have experimented with several approaches aimed at tackling the data-sparseness problem. In the following sections, we will present data-fabrication techniques, data-retrieval techniques, class-based n-gram approaches and class-based n-gram generalisation algorithms. We will also present our methods and experiments for handling multilingual language modelling.

In the interests of making the chapter self-contained, we briefly summarise a couple of key technical ideas. One very common type of LM is the *statistical n-gram language model*; this is an $(n-1)$th order Markov model, where the probability of one word in the sentence depends on the $n-1$ previous words. n is most often equal to 2, resulting in the *bigram language model*. In a bigram LM, the probability of one word w_n in the sentence, given the history of all previous words $w_1, w_2, \ldots, w_{n-1}$, depends only on the previous word:

$$p(w_n|w_1, w_2, \ldots, w_{n-1}) = p(w_n|w_{n-1}).$$

LMs are also frequently constructed by use of interpolation. An interpolated language model computes the probability of a word w, given the history of previous words (or word classes) H, as a linear combination of the probabilities given by two or more language models. In the case of two models, A and B, this probability can be expressed as

$$p(w|H) = \lambda \cdot p_A(w|H) + (1-\lambda) \cdot p_B(w|H),$$

where $p_A(w|H), p_B(w|H)$ are the two language models and λ is the interpolation weight between 0 and 1. Notice that λ can also be a function of H.

Table 16.1. Word-error rate with fabricated LM training data

System	Training data	WER (%)
Baseline	220 000 words	5.73
No LM	0	39.10
FabPhrase	1 300 000 words	14.66
FabSentence	1 500 000 words	12.49

16.2 Fabricating Domain-Specific Data

The data-sparseness problem for language-model training can be alleviated using grammars written by human language experts. It is well known that it is difficult to write a full-sentence grammar for a general domain, or even for a limited domain. However, our experience has shown that it is quite reasonable to assume that a grammarian can write a phrase grammar for a limited domain within a few weeks. The key advantage of a phrase grammar is that it can capture almost all the necessary information needed for the domain without having to exhaustively consider all the possible phrase combinations. For example, in the ATIS domain, it is quite easy to write down the most frequently used expressions for dates, times, costs, locations, transportation, meals, and arrival and departure events. Because phrase subgrammars are modular (Weng and Stolcke 1995), many expressions, such as dates, times, locations, and numbers, can be used across different domains. Therefore, moving into a new domain is even easier after phrase grammars have been written for some initial domains.

There are two potential ways to apply phrase grammars for language modelling. One is to use phrase grammars to generate random phrases and to train statistical language models, that is, to *fabricate* domain-specific data. The other is to embed phrase subgrammars in statistical language models, in a way similar to class-based language modelling. We will discuss in detail the second approach in Section 16.5.

We conducted a set of initial experiments in 1995, when we first took an English phrase grammar adapted from SRI's ATIS template matcher system (Jackson et al. 1991). This phrase grammar uses a vocabulary of 1 750 words, and about 200 non-terminals with approximately 3 600 rules. The majority of the rules are simply names (city names, airport names, etc.) with indicative words, such as preposition *from* for departure location and *to* for destination.

The SRI 1994 ATIS speech recognition benchmark system (Cohen, Rivlin, and Bratt 1995) was adapted for our real time SLT task. The resulting baseline

Table 16.2. Word-error rate with interpolated real training data

Real Data	Interpolated LM		Real Data Only LM
(# Words)	Weight	WER (%)	WER (%)
0	0.0	12.49	39.10
10 000	.9	7.75	9.14
20 000	.97	7.10	7.81
30 000	.97	6.65	7.12
40 000	.97	6.52	7.10
50 000	.95	6.27	6.89
100 000	.92	5.86	6.12
220 000	.97	5.69	5.73

speech recogniser has 5.73% word error rate with a bigram language model trained on 220 000 words of ATIS domain data[1]. Tab. 16.1 contrasts performance of this LM with bigram LMs trained on two artificial corpora. The first of these (FabPhrase) consisted of 300 000 phrases (1.3 million words) randomly generated using the phrase grammar. The second (FabSentence) has 100 000 fabricated sentences (1.5 million words) with in-vocabulary words randomly inserted in between the phrases. For completeness, we also conducted a set of experiments that linearly interpolated the FabSentence model with the models trained on different amounts of real data. The results are presented in Tab. 16.2.

Without using any real training data, the LM trained on 100 000 fabricated sentences led to an almost usable speech recogniser, which we found quite encouraging. A refinement, which we hope to investigate in the future, is to incorporate phrase tags in the tag set of the corpus. This could then be used to create a statistical n-gram tag model which in turn would be used to fabricate sentences. The idea is related to the class n-gram generalisation approach discussed in Section 16.5.

16.3 Better Use of Domain-General Data

Successful statistical language modelling requires large amounts of domain-specific data, which are not always available. As discussed above, direct use of widely available domain-general data often leads to unsatisfactory results,

[1] All the experiments in this section were carried out on the 1994 DARPA benchmark evaluation male test set, which contains 443 sentences and a total of 4 660 words.

due to vocabulary and style mismatches. One straightforward way to attack the problem (Iyer, Ostendorf, and Rohlicek 1994) is to interpolate an n-gram LM trained on small amounts of available domain-specific data with another n-gram LM trained on large amounts of domain-general data. By adjusting the weight of the domain-specific LM to a high value, one can reduce the effect of the mismatches in training data, but at the same time inherit robustness from the general LM.

By interpolating a domain-specific n-gram LM with a general n-gram LM, we utilise only local properties of the domain-general data, typically counts for bigram LMs. However, many of these local counts are irrelevant both to the domain and to the global constraints of the language. Including them therefore serves no useful purpose, and only degrades the quality of the LM. Instead of extracting n-gram counts directly from domain-general data, an alternative approach is to start by attempting to identify a subset of the general data related to the domain we are interested in, and extract counts only from that.

One of the assumptions of this approach is that so-called general data really consists of a mixture of data from many different specific domains. A strand of research in traditional Artificial Intelligence (AI) has for a long time advocated a script-based approach to language understanding (Schank and Abelson 1977), based on the assumption that shared human knowledge is organized in the form of scripts: routine procedures that people perform for different tasks and have an organisation that is well reflected in language. They are possibly related to the concept of domain organisation. Interestingly, our observations have several points of contact with script-based research.

With these ideas in mind, we designed a set of experiments in 1995. We selected some relevant news articles from Usenet news groups and the Switchboard corpus[2] LM data to conduct the experiments. To filter these corpora and retrieve relevant data, a distance measure was required. Adopted from information theory, perplexity is commonly used in language modelling as a measure of the fit between model and data. It is defined as:

$$Perplexity = 2^{-\sum_{x \in V_T} p'(x) * \log p(x)},$$

where $p(x)$ is the probabilistic distribution of the model, $p'(x)$ is the estimated probability from text data T, and V_T is the vocabulary of T. The lower the perplexity number, the better the model fits the data. Here, we use it as a distance measure for selecting sentences that are close to a given seed model.

[2] The Switchboard corpus is a conversational speech database collected over the telephone that includes a variety of conversation topics.

We first took articles from Usenet rec.travel.* news groups covering May and August of 1995, using the FabPhrase model from Section 16.2 as the seed model. Because many words in the general data were out of domain vocabulary (OOV), we only obtained 1 700 sentence fragments or chunks (as opposed to full sentences) totaling 11 000 words. We also tried using the Switchboard corpus as the domain-general data, hoping that the conversational styles and travel topics captured by that data could be helpful for the ATIS domain. The filtering process on Switchboard data yielded about 85 000 chunks, totaling 288 000 words. Two language models, which we will call *gnus*[3] and *swb*, were built using the two data sets just described.

We performed conventional interpolation of the models as baselines. As we can see in Tab. 16.3, the interpolated models gnus+FabPhrase and swb+FabPhrase both produce significant improvements over all the three individual models, and in particular over the FabPhrase model. We also combined all three language models (gnus+swb+FabPhrase) obtaining a further small improvement. All the recognition results are listed in Tab. 16.3.

These results show that using general-domain data can significantly improve an in-domain model. We also experimented with selective usage of general-domain data. A language model was built, that weighted more heavily (0.9) the low perplexity fragments from the general-domain (Switchboard and gnus) data extracted using the FabPhrase model: we called it *gnuswb.median*. This LM was then interpolated with the fabricated data to yield yet another model, gnuswb.median+FabPhrase. The gain from selective usage of general-domain data was smaller than expected, and we attribute this largely to our simplistic use of perplexity as a distance measure for retrieving relevant sentence fragments. For example, chunks ending with “the”, such as “I was just wondering if the”, and “when checking into the originating flight and the”, are quite frequent, and this distorts the perplexity of relevant phrases in the general data that end with “the”. Hence, some preprocessing is required in order to make this distance measure more effective. Alternative distance measures also need to be investigated, including labeling sentences with phrase grammars and preferring ones with high phrase label coverage. Another possible reason for this smaller-than-expected gain can be the fabricated seed model, FabPhrase. As explained before, FabPhrase is trained on randomly generated phrases from the phrase grammar. Therefore, this model does not fully reflect the real probabilistic distribution of any real in-domain data, especially in the case of phrase boundaries. An improvement would be to use the technique

[3] gnus is the name of the software used to access Usenet.

Table 16.3. Word-error rate with interpolated real training data.

Model	WER (%)
gnus (40K words)	30.19
swb (1.5M words)	18.88
FabPhrase	14.66
gnus+swb	14.76
swb+FabPhrase	11.57
gnus+FabPhrase	10.79
gnus+swb+FabPhrase	10.41
gnuswb.median+FabPhrase	10.17

proposed at the end of Section 16.2 to build a seed model.

16.4 Unsupervised Language Model Adaptation

When there is not enough domain-specific data, a possible solution is to perform *unsupervised adaptation* of the language model. In unsupervised adaptation, the language model is reestimated on in-domain speech data without human transcriptions, using a recogniser to obtain automatic, albeit imperfect, transcriptions.

We thought it would be interesting to attempt to do this using the gnuswb.median+FabPhrase LM from the previous section; recall that this language model was trained entirely on non domain-specific data, with the phrase grammar as its only domain-specific knowledge. To investigate dependence on the quantity of adaptation data used, we selected the 1994 DARPA ATIS male development (d1994m) and 1993 DARPA ATIS male evaluation test sets (e1993m) as adaptation, in addition to the 1994 DARPA ATIS male evaluation test set (e1994m). A first recognition pass on e1994m, d1994m, and e1993m was performed to obtain automatic transcriptions for the three data sets. After this, the three automatically transcribed data sets, hyp-e1994m, hyp-e1993m, and hyp-d1994m, were incrementally merged to form three adaptation data sets, hyp-e1994m, hyp-(e1994m+e1993m) and hyp-(e1994m+e1993m+d1994m), with 4 654, 9 210 and 13 541 words, respectively. These were then used to build three models for adaptation. All three models were interpolated with the gnuswb.median+FabPhrase model, using a weight of 0.9 for the three models created from adaptation data. Three new recognition systems with the three corresponding unsupervised adapted models were tested on e1994m data.

The results are shown in Tab. 16.4. The best version, hyp-(e1994m+e1993m-

Table 16.4. Word-error rate with unsupervised adaptation

Unsupervised Adapt. Data	WER (%)
none	10.17
hyp-d1994m	9.08
hyp-e1994m	9.01
hyp-(e1994m+e1993m)	8.78
hyp-(e1994m+e1993m+d1994m)	8.76

+d1994m), displays a 13.9% relative improvement over the baseline system, using 13 541 words of unsupervised adaptation data. With less than 5 000 words of unsupervised adaptation data, whether on the same data set (hyp-e1994m) or another data source (hyp-d1994m), we still obtained more than 10% relative improvement. However, additional iterations of unsupervised adaptation did not improve the recognition performance.

Our significant improvements in word-error rates can be explained by two reasons. First, the baseline system incorporates a variety of grammatical instances from the phrase grammar and selected data, and, although the distribution of the fabricated data is distorted, this LM is a satisfactory starting point. Second, this relatively low word-error rate baseline system is used in unsupervised mode to create reliable hypotheses, which are weighted heavily during unsupervised adaptation.

16.5 Class-Based Language Models

There are two extreme sets of approaches in language modelling. The first is to model flat word strings using statistical n-gram, techniques, without considering the internal structure of sentences. This approach has the merit of simplicity: it is easy to implement, and it is easy to train robust models. On the other hand, because it does not take internal structure into consideration, it will either not model long-distance dependencies when n is small, or we will face a data-sparseness problem when n is big. The second approach is to model whole sentence structures. This approach has the merit of precision: it provides detailed information and improves predictability. It does, however, suffer from a severe data-sparseness problem, making it very hard to train such models.

As a compromise approach, class-based n-gram language modelling has been a very successful technique that can be used when the amount of available

Table 16.5. Word-error rates of word bigrams vs. class bigrams with respect to different amounts of data in English ATIS

LM	Training (words)	WER (%)
word	220 000	7.02
word	150 000	7.38
word	100 000	7.73
word	50 000	8.26
class	220 000	6.91
class	150 000	7.10
class	100 000	7.40
class	50 000	7.40
word+class	220 000	6.37
word+class	100 000	7.10
word+class	50 000	7.21

data is limited (Brown et al. 1992). A more flexible approach, which we will talk about later in this section, is to model important phrases in a sentence. In our SLT work, we implemented a class-based bigram LM and tested it with our English and Swedish ATIS systems.

The class-based bigram LM is defined as:

$$P(w_2 \mid w_1) = \sum_{(C_1, C_2), where\ w_2 \in C_2, w_1 \in C_1} P(C_2 \mid C_1) * P(w_2 \mid C_2),$$

where C_1, C_2 are the classes of words w_1, w_2, respectively. Notice that the class-based bigram LM is essentially an HMM (see Chapter 14) with the state-transition probability being the class bigram ($P(C_2 \mid C_1)$) and the output distribution being the class-membership distribution ($P(w_2 \mid C_2)$). In our experiments, the English classes were created manually; they include city names, airlines, airline codes, and so on. The Swedish classes were manually translated from English ones with some minor modifications.

Tab. 16.5 lists the recognition word-error rates for the English ATIS system with different amounts of language-modelling training data, using word-bigram (*word*), class-bigram (*class*), and interpolated (*word+class*) language models. The test set was again the 1994 DARPA benchmark evaluation male data.

The results of Tab. 16.5 show that with larger amounts of training data (220 000 words), pure class-based bigram LMs perform as well as word-bigram LMs, while with small amounts of training data (50 000 to 100 000 words),

Table 16.6. Perplexities of word-bigram model, class-bigram model, and interpolated models for English, with OOV rate being 0.1%

	Perplexity
word	33.0
class	25.4
word+class	24.6

Table 16.7. Word-error rates and perplexity of word-bigram model, class-bigram model, and interpolated models for Swedish on 444 sentences (3 758 words)

	WER (%)	Perplexity
Baseline	7.90	40.260
class	8.01	22.208
word+class	7.64	20.715

pure class-based bigram LMs outperform word-bigram LMs significantly. The interpolated word-bigram and class-bigram systems gave further improvement over the pure class-bigram systems. The results also show that the interpolated systems perform as well as the word-bigram systems that are trained with twice as much data. Compared with existing recognisers that have word-bigram LMs trained with large amounts of data, the new English system with class-based bigram LMs mixed with word bigram LMs reduced the WER from 7.02% to 6.37% on a test set of 4 660 words, a significant improvement. The perplexity improvements of the class-based and interpolated LMs over the word-bigram LM using the 220 000 words of training data are consistent with the word-error rate reductions (see Tab. 16.6).

We repeated a subset of experiments on our Swedish system, and obtained slightly better perplexity reduction, but only slight word-error rate reduction (see Tab. 16.7). This difference is probably due to the limited variability of our Swedish test set.

Class n-gram LMs allow convenient combination of hand-coded linguistic knowledge (in the form of class definitions) with statistical parameters trained from data (the class n-gram probabilities). In structuralist terms, class-based LMs improve generalisation along the *paradigmatic* dimension: substitutions of known words into contexts they haven't been previously observed in. In the remainder of this section we will investigate a way to improve the generalisation of n-gram models along the *syntagmatic* dimension, by inferring n-grams that were not observed in training, but are plausible given the observed ones. While the technique can be applied to word- or class n-grams, it makes most sense in combination with class n-grams, leveraging the generalisation afforded by the word classes.

At the core of our technique for n-gram generalisation is an algorithm that was orignally developed for compressing word lattices generated by speech recognizers (Weng, Stolcke, and Sankar 1998). The idea is to merge nodes in a lattice based on shared left or right contexts. Nodes with identical or largely overlapping contexts are inferred to be instances of the same underlying grammatical context and combined, resulting in a more compact representation. The version of the algorithm that considers right contexts is given below:

Backward lattice reduction algorithm Let $S_{out}(n)$ be the set of successor nodes of node n. Let $word(n)$ denote the word name of lattice node n.

For each lattice node *n* in reverse topological order (starting with the final node):

- for each pair of predecessor nodes (i, j) of node *n*:
  - if $word(i) = word(j)$ and $\frac{\|S_{out}(i) \cap S_{out}(j)\|}{\| \min\{S_{out}(i), S_{out}(j)\}\|} \geq r_{min}$ (or, $word(i) = word(j)$ and $\frac{\|S_{out}(i) \cap S_{out}(j)\|}{\| \max\{S_{out}(i), S_{out}(j)\}\|} \geq r_{max}$), then merge nodes i and j.

In the algorithm, r_{min} and r_{max} are context overlap ratios to be adjusted empirically. Setting $r_{min} = 1$ (or, $r_{max} = 1$) forces the two outgoing node sets to be the same; this variant is called the *exact reduction algorithm*, and would preserve the set of strings represented by the lattice, i.e., give no generalisation. When $r_{min} < 1$ or $r_{max} < 1$, the algorithm generalises the coverage of the lattice by allowing additional transitions. For example, before the reduction process, as shown in Figure 16.1a, word string *ead* is not in the lattice; after the reduction process (Figure 16.1b), the word string *ead* is included. The decision to include word string *ead* in the lattice relies on the overlap of the existing word strings. In our example, word strings *...eab...* and *...fab...* overlap after *ab*, and word strings *...fab...* and *...fad...* overlap before *fa*. Therefore, it is reasonable to assume that *...ead...* is a valid string. Intuitively, the reliability

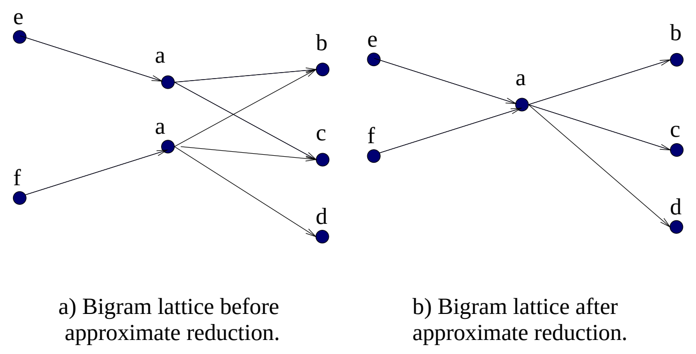
e
a
b
a
c
f
d
e
b
a
c
f
d
a) Bigram lattice before approximate reduction.
b) Bigram lattice after approximate reduction.

figure 16.1. Illustration of the reduction algorithm.

of the new word string(s) correlates with the overlap ratios. The higher the proportion of overlap between two sets of strings, the more confident we can be that the new word string(s) are grammatical.

Instead of testing the overlap ratios of the outgoing node sets, the algorithm can be executed equally well by requiring minimal overlap ratios for two incoming node sets. This gives the forward version of the reduction algorithm. Forward and backward versions can be performed several times in alternation to locate new merging opportunities created by the respective other processing direction.

We can embed the lattice reduction algorithm in a new algorithm that generalises word-class sequences obtained from a training corpus. The generalised lattice representation is used to generate a new set of n-grams (including ones not observed in training), from which a new n-gram LM can be constructed. As a final tuning step, we interpolate the resulting n-gram model with the word n-gram model obtained directly from our original training data. This counteracts possible overgeneralisation and ensures that the overall model is no worse than the standard word-based LM. The complete procedure is as follows:

Class n-gram generalisation algorithm:

1. Automatically tag the training data with a (hand-written) class or phrase grammar. This yields a new training corpus in which defined words and phrases have been replaced with class names. Where multiple tag sequences are possible, select the one that replaces the largest number of words with class/phrase tags.
2. Combine the tagged training sentences in parallel, so as to form a large

lattice of disjoint paths, sharing only the same initial and final nodes.
3. Apply the lattice reduction algorithm to the lattice thus constructed.
4. Generate a large sample of word sequences from the reduced lattice, by combining random walks through the lattice with randomly generated word sequences for the class labels encountered.
5. Train a word n-gram LM from the artificial sentence sample.

We tested this algorithm using a small set of training sentences from the English ATIS corpus, comprising about 20 000 words; the test set was, as before, the 1994 DARPA male evaluation data. Classes of words and phrases were predefined by a hand-written finite state grammar consisting of about 144 nonterminals and approximately 4 000 rules. The majority of the rules defined proper name classes, such as city and airport names, with some additional classes for numbers, dates and similar productive phrase types. An important difference to the previous experiments with class-based LMs was the inclusion of the latter types of classes, which generate an infinite set of word sequences.

Both the approximate (generalising) and the exact lattice reduction algorithms were applied to the lattice built from this data. The overlap ratio threshold for generalising (r_{min}) was 0.5, i.e., if lattice nodes shared more than half their predecessors or successors they were considered mergeable. The exact approach gives us a baseline with which to compare the effect of generalisation. Since the set of word and class n-grams is not generalised, the results should be comparable to that of a standard class n-gram LM in this case. To build the generalised n-gram model we generated random sentences from the lattice, totalling about 1 500 000 words. Finally, the generalised n-gram LM was interpolated with a standard word n-gram model trained from the same original 20 000 word corpus.

The interpolation weight was set in a jack-knifing procedure: the data was first split in two halves, the algorithm was executed separately on each half, and the interpolation weights optimised on the respective other half (by minimising perplexity). The algorithm was then run on the full training data, using the average of the two previously optimised interpolation weights.

Tab. 16.8 shows the results obtained, for both bigram and trigram LMs. For comparison, the performance of various previously discussed LMs trained on the same amount of data are also given: the simple word-based LM, the interpolated class-based LM (as described earlier in this section) and the interpolated LM based on fabricated data (see Section 16.2).

We see that the lattice-based models perform well compared to all the baseline models, especially for the trigram case. The bigram LM using fabricated data performs somewhat better than the other bigram LMs, but it should be

Table 16.8. Results of generalized n-gram LMs and various baseline models, trained on 20 000 words of ATIS data.

	WER (%)	
Model	Bigram	Trigram
Word n-gram	7.81	7.96
Word+Class n-gram	7.51	7.12
Word+FabSentence	7.10	6.82
Word+Exact lattice	7.38	6.57
Word+Generalized lattice	7.40	6.46

noted that that model makes crucial use of a hand-written phrase grammar and is thus not directly comparable. Compared to the class-based LM, the model generated from the exact lattice shows some improvement, which can be attributed to the more productive phrase-class definitions. Finally, for the trigram case, we see an additional improvement over the exact-lattice approach, which we attribute to the generalising effect of the approximate lattice reduction.

While the observed improvements remain small, we see the class-based n-gram generalizaion technique presented here as a promising new tool to improve LMs trained from very little raw training data. Future work will have to address additional research questions and potential improvements. These include the question of how to find optimal parameters to control the generalisation algorithm, and how to best combine the method with the other techniques presented in this chapter.

16.6 Multilingual Language Modelling

In today's world, a significant number of people speak multiple languages, and *code switching* is becoming more pervasive. Code switching is used to refer to situations where people switch languages within sentences. This can be a very frequent phenomenon in multilingual systems like SLT. Despite its growing significance, code switching is not well studied in language modelling and speech recognition. We have carried out some initial investigations of this problem in the context of our Swedish/English bilingual recognition system (Weng et al. 1997). We discuss the acoustic modelling aspects of multilingual systems in Chapter 18. In this section we present two LM approaches that were tested in the bilingual recogniser.

The first approach pools together the Swedish language model data with the

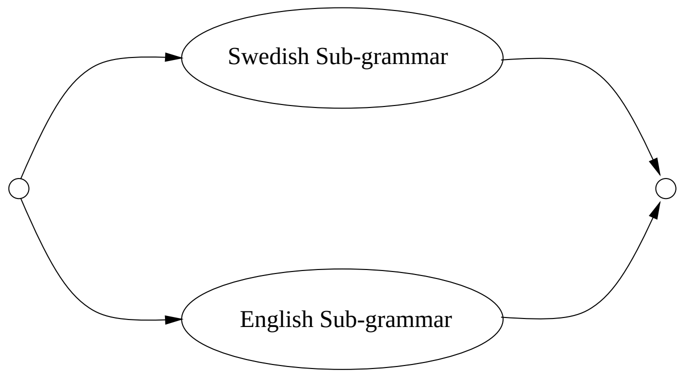

figure 16.2. Illustration of the constrained LM.

English one. A bigram language model (called *shared LM*) was built on the pooled data. Because there are no code-switching instances in the training data, Swedish words can only be followed by Swedish words and English words can only be followed by English words. Transitions between the words of the two languages in the bigram language model can however still be realised through its *backoff node*: in bigram language models, probabilities for word pairs that have no (or very few) occurences in the training data are approximated by appropriate weighting of the unigram probabilities, and this can be efficiently implemented through a backoff node (Katz 1987).

A second approach uses a *constrained LM* that does not allow any transition between the two languages, except in the initial state (Figure 16.2). When tested on English data, the recognition system with the constrained LM performs almost the same as the one with the shared LM. However, when tested on Swedish data, the constrained LM performs significantly better than the shared LM. More details about these experiments, including different types of acoustic models, can be found in Chapter 18.

The above approaches represent two extremes: one allows transitions at any time, and the other allows no transitions at all. Motivated by this study and our work on phrase-based LMs, we plan to pursue approaches that exploit some of the advantages of the extreme positions without inheriting all of their weaknesses. One possibility is to allow proper names to be switchable among different languages. Among others, these proper names would include personal names, location names (cities, airports, etc.), and names of companies and products. A more flexible and generic approach is to allow language switching between phrases, which can be done using the phrase-based language modelling techniques discussed in Section 16.5. Unfortunately though, it is difficult to evaluate these approaches empirically without a multilingual corpus that includes code-switching phenomena. We see collection of such a corpus as an important task for future work.

16.7 Conclusions

In specialised application domains and in multilingual environments, as in SLT, acquiring a corpus to train the language model is a serious task involving non trivial design problems, as we saw in Chapter 8. In many practical applications, we are faced with the problem of building a language model using little or no training data. For a multilingual environment, additional issues, such as code switching, need to be properly addressed. These have been the main foci of our language modelling effort in SLT.

We dealt with the first problem in a number of different ways, including: (1) various applications of hand-written phrase grammars, such as fabricating domain-specific data and class-based n-gram generalisation; (2) automatic extraction and selection of relevant phrasal components from readily available data for training; and (3) unsupervised adaptation of the language model. Some of the approaches have led to very encouraging results. Others also show certain promise, or point to new directions.

With regard to multilinguality, we examined different issues in the construction of multilingual language models for our recognition systems, proposed and experimented with a few different approaches, and obtained interesting results, especially for code-switching phenomena.

We hope that our continuing LM research effort in SLT contributes to the maturity of speech technology and its increasing application to multilingual interaction in the world.