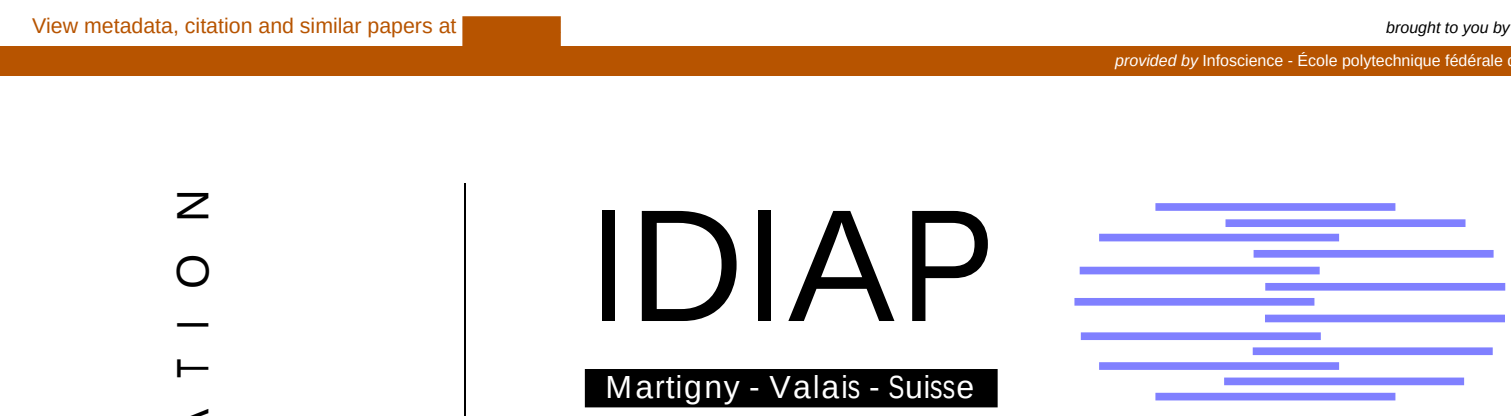
IDIAP COMMUNICATION



Baseline System for Hybrid Speech Recognition on French (Experiments on BREF)

Johan M. Andersen

IDIAP–Com 98-7

December 1998

Dalle Molle Institute for Perceptual Artificial Intelligence • P.O.Box 592 • Martigny • Valais • Switzerland

phone +41 – 27 – 721 77 11
fax +41 – 27 – 721 77 12
e-mail secretariat@idiap.ch
internet http://www.idiap.ch

a Dalle Molle Institute for Perceptive Artificial Intelligence PO Box 592 CH-1920 Martigny, Switzerland

Abstract

This report describes work done at IDIAP within the THISL project towards improving a French baseline system. This work has focused on lexicon training (described elsewhere), language model (LM) training, and training of multi-layer perceptrons (MLPs).
MLP training and recognition experiments were performed on the French speech database BREF containing read clean speech of studio quality, and LMs were trained on 1990-92 of the journal "Le Monde". In general the differences in word error rate (WER) are relatively small.
The initial system had WER=28.3%, which improved to 27.3% when a more careful LM text preprocessing was applied, using a trigram of equal size and a previously trained MLP with 1000 hidden units (HU). It was realized that some of the BREF text was chosen from "Le Monde", January 1990. Leaving out this month increased WER to 28.9%.
Careful training of a larger MLP (2000 HU) improved recognition to 25.9% WER. When this best MLP was used with less pruning, and replacing the trigram with a 4-gram, the best performance was observed: 23.3% WER.
A large number of different LMs and MLPs were trained and tested in this work, and various settings of decoding parameters were tested. All these experiments revealed many research directions for the future, that are listed at the end of this document.
The work presented here is considered to be improvements of the baseline system, and is thus a starting point for further work, applying new and more research oriented methods in language modeling, lexicon training, and acoustic modeling.

Contents

Chapter 1

Introduction

This report describes work done at IDIAP in 1998 towards improving the French baseline speech recognition system. The work was done within the European THISL project, where it was clear that the French speech recognition system needed improvement.

Several improvements have been performed: Training of a pronunciation dictionary, which has been described elsewhere, and then the focus of this report: training of language models and training of neural networks (MLPs) for the acoustic modeling, as well as tuning of recognition parameters.

The report starts with a short survey over relevant theory, and a brief description of the software packages used within this work. The main work has been organized into three parts: LM training, MLP training, and recognition experiments. At the end there is an outline of how this work can be continued and completed.

1.1 Acknowledgments

IDIAP will like to thank FPMs for allowing us to compute language models at their computer facilities. Dr. Olivier Deroo and Mr. Christophe Riss were particularly helpful in this work.

We also owe thanks to FPMs for providing the MLP that was the starting point for this work, as well as a forced alignment that was used in an MLP retraining.

Thanks also to Mikko Kurimo for making comments on an earlier draft of this report.

Chapter 2

Theory

All experiments reported in this document were performed in the context of hybrid speech recognition, where "hybrid" here means a mixture of hidden Markov models (HMMs) and artificial neural networks (ANNS). In all cases the ANN is a multilayer perceptron (MLP) with one hidden layer.

A summary of the theory behind hybrid systems will be presented here to allow a better understanding of the experiments described in later sections.

Bayesian decision theory says that the optimal (min. sentence error rate) decoding consists of choosing the word sequence $\hat{W}$ with the highest a posteriori probability:

$$\hat{W} = argmax_W P(W|X) \tag{2.1}$$

The a posteriori probability can be split up in the following way:

$$\begin{aligned} P(W|X) &= \frac{P(X|W)P(W)}{P(X)} && (2.2)\\ &= \frac{P(W)\sum_Q P(X,Q|W)}{P(X)} && (2.3)\\ &= \frac{P(W)\sum_Q P(X|Q,W)P(Q|W)}{P(X)} && (2.4)\\ &= P(W)\sum_Q \frac{P(X|Q)}{P(X)}P(Q|W), && (2.5) \end{aligned}$$

where W is a sequence of words, X is a sequence of acoustic vectors (often represented by feature vectors), and Q is a sequence of hidden states in the Markov model. In the above X is assumed independent of W given Q.

In the decoding $P(X)$ can be left out because it is the same for all hypotheses W. $P(W)$ is the probability for the word sequence, and is computed by the language model. $P(X|Q)$ is the probability for the acoustic sequence given the state sequence. And $P(Q|W)$ is the pronunciation probabilities, that is the probability for state sequence given a word sequence.

2.1 LM probabilities

The LM probability, $P(W)$, is the a priori probability for the word sequence, and is computed by the language model. The language models tested in this work, are bigrams, trigrams and 4-grams, that assumes 1st, 2nd or 3rd order Markov properties. That is, for a bigram:

$$P(W) = P(w_1, w_2, \ldots, w_n) \tag{2.6}$$

$$= P(w_1)\prod_{i=2}^{n} P(w_i|w_1, w_2, \ldots, w_{i-1}) \quad (2.7)$$

$$= P(w_1)\prod_{i=2}^{n} P(w_i|w_{i-1}), \quad (2.8)$$

and for a trigram:

$$P(W) = P(w_1)P(w_2|w_1)\prod_{i=3}^{n} P(w_i|w_{i-2}, w_{i-1}), \quad (2.9)$$

In a trigram, a word tuple (w_1, w_2) or (w_1, w_2, w_3) might, or might not have its own entry in the LM. If it has its own entry, the LM will contain the conditional probability $P(w_2|w_1)$ or $P(w_3|w_2, w_1)$ respectively.

If a word-triple (in the case of a trigram) is missing in the LM, we still need to be able to compute $P(w_i|w_{i-2}, w_{i-1})$ in (2.9). This is done by "backing off" to bigrams if there is no trigram, and if there is no bigram neither, we back off to unigrams (which are available for all words in the LM).

For this reason, word pairs and single words in the LM also have a number called a back-off weight: $b(w_1)$ or $b(w_1, w_2)$, which is used to assure that given a context (w_1, w_2) all probabilities $P(w_3|w_1, w_2)$ sum to one (summed over w_3).

Let $m(w_1, w_2, w_3)$ (m for model) describe conditional probabilities $P(w_3|w_1, w_2)$ for word triples that have their own entry in the LM, and similar for word pairs. Then the back-off from trigram to bigram is done in the following way:

if $m(w_1, w_2, w_3)$ exists:

$$P(w_3|w_1, w_2) = m(w_1, w_2, w_3) \quad (2.10)$$

else if $b(w_1, w_2)$ exists:

$$P(w_3|w_1, w_2) = b(w_1, w_2)P(w_3|w_2) \quad (2.11)$$

else:

$$P(w_3|w_1, w_2) = P(w_3|w_2) \quad (2.12)$$

Similarly the back-off from bigrams to unigrams is done by:

if $m(w_1, w_2)$ exists:

$$P(w_2|w_1) = m(w_1, w_2) \quad (2.13)$$

else:

$$P(w_2|w_1) = b(w_1)m(w_2) \quad (2.14)$$

2.2 Acoustic probabilities

If $P(X)$ is left out, we only need to calculate the probability $P(X|Q)$ of an acoustic sequence given a state sequence. $P(X|Q)$ can be split up into a product, if it is assumed that acoustic vectors x_i and x_j are independent for two different time indices i and j given the hidden state for those frames:

$$P(X|Q) = P(x_1, x_2, \ldots, x_T|q_1, q_2, \ldots, q_T) \quad (2.15)$$

$$= \prod_{i=1}^{T} P(x_i|q_i) \quad (2.16)$$

The above standard HMM assumption is very crude, especially as the x_i that is used in the hybrid system is a vector composed of features from 4 frames to each side. This is probably the most important reason why acoustic scaling is needed.

In theory the MLP estimates a posteriori probabilities for the different phonemes/states, and can be converted into (scaled) likelihoods by dividing by priors:

$$\frac{p(x|q)}{p(x)} = \frac{p(q|x)}{p(q)}, \tag{2.17}$$

which can be used in (2.16).

Another way to factorize $P(X|Q)/P(X)$ is:

$$\begin{aligned}
\frac{P(X|Q)}{P(X)} &= \frac{P(Q|X)}{P(Q)} && (2.18)\\
&= \frac{P(q_1, q_2, \ldots, q_T|x_1, x_2, \ldots, x_T)}{P(q_1, q_2, \ldots, q_T)} && (2.19)\\
&= \frac{\prod_{i=1}^{T} P(q_i|x_i)}{\prod_{i=1}^{T} P(q_i)} && (2.20)\\
&= \prod_{i=1}^{T} \frac{P(q_i|x_i)}{P(q_i)}, && (2.21)\\
& && (2.22)
\end{aligned}$$

which arrives at the same result. It is seen that states q_i and q_{i+1} are assumed independent given the acoustics x_i and x_{i+1}. This might be close to the truth if the acoustic (feature) vectors contain information about neighboring frames. However, the above derivation also assumes that states q_i and q_{i+1} are independent, which is a clearly false assumption. The factorization of $P(Q)$ might be better, if a first order (e.g.) Markov assumption was used, which would mean that MLP output should be divided by a priori state transition probabilities in stead of just prior probabilities as is done currently (see (2.17)).

2.3 Pronunciation probabilities

$P(Q|W)$ can be called a pronunciation probability because it is the probability for a state sequence Q given a word sequence W. The pronunciation dictionary together with the phoneme HMMs are used to calculate $P(Q|W)$.

The dictionary contains one or more transcriptions for each word in the vocabulary possibly together with a prior probability for each pronunciation if there are more than one. If there are no priors, all pronunciations for a word are assumed to have equal prior probability.

The phoneme HMMs contain self-loop and transition probabilities.

Both the pronunciation dictionary including pronunciation priors, and basic phoneme HMMs can be trained with the dr_embed program provided by ICSI.

2.4 LM/acoustic scaling

If "true" probabilities were available, (2.1) and (2.5) would provide the optimal decoding. However, we only have probability estimates, coming from the MLP, a LM, phoneme HMMs, and a pronunciation dictionary. Some of these probabilities might be over-estimates or under-estimates.

$$\begin{aligned}
P(W|X) &= P(W) \sum_{Q} \frac{P(X|Q)}{P(X)} P(Q|W), && (2.23)\\
&\approx \hat{P}(W)^{\text{l.s.}} \sum_{Q} \left(\frac{\hat{P}(X|Q)}{\hat{P}(X)} \right)^{\text{a.s.}} \hat{P}(Q|W)^{\text{p.s.}}, && (2.24)
\end{aligned}$$

Language scaling (l.s.) and acoustic scaling (a.s) are two parameters that can be tuned, and allow a scaling of the importance of the different components in (2.24). A third scaling, pronunciation scaling (p.s.) could be added for completeness.

Chapter 3

Software packages

3.1 STRUT ASR software

The STRUT software was only used in an indirect way. The RASTA-PLP features for the training and test set was calculated with STRUT in the first place. Since this work used the ICSI software (qnstrn) for MLP training, the STRUT files were converted into pfiles needed for qnstrn.

Note that the STRUT version of PLP-RASTA is slightly different from the ICSI version. STRUT has an additional pre-weighting function,

$$1 - az^{-1} \qquad (a \text{ defaults to } 0.95), \tag{3.1}$$

and the (fixed) coefficients to calculate delta parameters in the ICSI RASTA program are

$$\Delta\text{-kernel:} \qquad [-4,-3,-2,-1,0,1,2,3,4] \tag{3.2}$$

$$\Delta\Delta\text{-kernel:} \qquad [-2,-1,0,1,2] * [-2,-1,0,1,2] \tag{3.3}$$

$$[-4,-4,-1,4,10,4,-1,-4,-4], \tag{3.4}$$

which is slightly different from what was used with STRUT:

$$\Delta\text{-kernel:} \qquad [-2,-1,0,1,2] \tag{3.5}$$

$$\Delta\Delta\text{-kernel:} \qquad [2,1,-2,-2,-2,1,2]. \tag{3.6}$$

Since the Δ-kernels and $\Delta\Delta$-kernels are specified by the user in STRUT, it would be good in the future to use default values that are the same as in the ICSI program, to make them compatible (if it doesn't make any difference in performance).

Also, if it does not increase WER, a in the STRUT pre-weighting should be set to zero, still to enhance compatibility.

3.2 Lexicon training program: babylex

Babylex is a program for training a pronunciation dictionary. From forced alignments, it calculates prior probabilities for the different pronunciations in a dictionary. It also introduces word-dependent minimum-durations for the different phonemes. To allow a better generalization, it is possible to train probabilities for a set of phonological rules that the user specifies. These rules can then be applied to words that didn't occur in the lexicon training data.

The use of babylex is not described in this report, as it has already been documented by Dan Gildea who trained the lexicon, which has been used in this work.

3.3 MLP training program: qnstrn

qnstrn is a version of the ICSI software for MLP training. The core of this software is the same that is found in STRUT. The "qn" stands for QuickNet, which was developed at ICSI to allow fast training of MLPs. It is highly vectorized, and exists in a version for Sun/UNIX and a version that run in fixed-point on the SPERT-board.

This last feature is important because the training is faster on SPERT board, especially for big MLPs. It runs 3-9 times faster than a 167 MHz Ultra SPARC; for 300 MHz Ultras, the performance has not been measured, but the advantage of the SPERT of cause is smaller, but still might run 2-5 times faster. And then workstations are often loaded with other processes that slow them down.

Indeed, the size and amount of MLP training experiments could not have been done without using the SPERT boards, since in some cases it might take more than a week to train an MLP, even on SPERT.

3.4 dr_embed embedded training script

dr_embed is a script that embeds lexicon training and MLP training and forced alignment in an iterative process.

3.5 Y0 decoder

The Y0 ("Why Not") decoder was only used as part of dr_embed to do forced alignment (it can be used for recognition with medium size vocabulary ($\tilde{1}000$ words) as well). Given local phoneme posteriors from a (trained) MLP, a pronunciation dictionary, phoneme HMMs, and a feature file, it creates a forced alignment, that can be converted into a label pfile and used as target for MLP training.

3.6 noway LVCSR decoder

The noway decoder was used in all ASR experiments reported here. noway takes an n-gram, a pronunciation dictionary, a phoneme HMM file, and local state posteriors. Using a beam-search, the decoder combines all these information sources into a word sequence hypothesis. noway also allows a lattice output, which has not been used in this work.

The n-gram must be in ARPA format or its own binary format (no description available) which is more compact. To save disk-space, and speed up loading in the LM, noway can convert ARPA format into its binary format before-hand.

The phoneme HMM file specifies all phonemes, how many states they contain each, what transitions that are allowed, and with what probabilities. It also ties each state to one of the output from the MLP. In this work, there is one MLP output per phoneme, but the phoneme file contains several HMMs for each phoneme: one for each minimum duration as previously determined by the babylex program.

3.7 wordscore program

To determine the word error rate (WER), the ICSI wordscore program was used. Before passing the word sequence hypothesis from noway to the wordscore program, all '_' and '-' were replaced with spaces in both the hypothesis file and the reference file. This preprocessing was originally introduced to allow a fair comparison to multi-words, where word tuples are concatenated with underscores. Also, we didn't want "vingt-et-un" and "vingt et un" to be different when calculating word errors.

3.8 CMU/Cambridge LM Toolkit

All the LMs were generated with the CMU/Cambridge LM Toolkit. It is a collection of programs and scripts, that takes a text-source as input, and calculates various statistics, and ultimately produces a back-off n-gram. It has its own compact binary format (different from the noway binary format!!) and can convert this format to ARPA format. It is thus convenient to use the binary format to store and transfer big LMs, that can then be converted to ARPA format and further to the binary noway format, when the LM has to be used.

The LM Toolkit has its own well-written html documentation, which can be consulted for more detailed information. This software is freely available from Cambridge University.

Chapter 4

The pronunciation dictionary

In all the recognition experiments, the same dictionary was used. And the same dictionary was also used for the forced alignment in the MLP retraining. This dictionary was produced by Dan Gildea from ICSI during a visit at IDIAP June-July 1998. This work was originally sponsored by the SPRACH project but fits perfectly in with the THISL project as well.

4.0.1 The phoneme set

The phoneme set that was used is a combination of the MBROLA[1] phoneme set and the LIMSI[2] phoneme set. It consists of 46 phonemes: 35 basic phonetic units that correspond to 35 of the MLP outputs, four silence models that correspond to the last MLP output, and finally seven of the basic units but with a possible skip. These seven phonemes are used for the liaisons. All phone models had a left–to–right structure.

However, during the lexicon training these 46 phoneme HMMs were expanded to many more HMMs with different minimum durations.

[1] For more details compare http://tcts.fpms.ac.be
[2] For more details see http://www.limsi.fr/Recherche/TLP/PageTLP.html

Chapter 5

Language model training

5.1 The "Le Monde" text database

The language models described in this document were all trained on the French journal "Le Monde". This work was partly performed at Faculté Polytechnique de Mons (FPMs). A total of 10 years of the journal is available on CD-ROM, from which it is possible to dump a raw text version with some header information for each article.

For historical reasons, only three years were used: 1990, 1991 and 1992. As will be pointed out later, the use of all 1990 is a problem because the text for the BREF database was partly selected from January 1990 of Le Monde. This means that some of the BREF test (and training) data are in the LM training set. This situation was later corrected, by first attempting to remove the "BREF sentences" from the LM training set, and later by leaving all of January 1990 out of the LM training set.

After pre1 preprocessing, the years 1990 – 1992 of the "Le Monde" database contains 66 million words (378 thousand different words) in 2.5 million sentences in 718 thousand paragraphs.

The most frequent words are shown in figure 5.1. The words $< \cdot >$ are markers inserted in the text to indicate the beginning of a sentence ($< s >$) and the end of a sentence ($< /s >$) and similar for paragraphs ($< p >$, $< /p >$) and articles ($< art >$, $< /art >$).

It can be seen that the word "de" in average appears more than once (1.28 times) in each sentence, which is not very surprising.

One also remarks the words "président", "état", "politique", "premier", and "ministre", which all indicates that the source text is often concerned with politics. The relative high frequency of "pourcent" and "virgule" tells that numbers/quantities are frequent.

Figure 5.2 shows statistics for n-grams in the text database "Le Monde" 1990-92. It shows how many n-grams occurred 1 time, 2 times, ..., 50 times.

Except for the n-grams only occurring 1, 2 or 3 times, the curves describe straight lines in the double logarithmic plot. This implies a curve of the form $y = ax^b$, where the coefficients can be read of the graph to be approximately ($a = 2 * 10^6, b = -1.8$) for bigrams.

The curve for bigrams is flattest and the curve for 4-grams is the steepest, which means that compared to bigrams, there are relatively more 4-grams occurring only once. This can be interpreted as an indicator of a text database that is too small. Ideally, if one had "enough" training data (say 10 or 100 or 1000 times more), one would expect a bell-shaped curve (where the tail might still be described by $y = ax^b$). This is so because if one is to reliably estimate the least likely n-gram, they have to occur a certain number of times. So if the curve is not bell-shaped, there are many n-grams occurring only a few times, which means that the probability estimates for these n-grams become un-precise.

However, maybe it is not likely to ever obtain a bell-shaped curve because typos continuously introduce "new" words and thereby "new" n-grams. Also, the distribution has to be the same throughout

de 3186843	qui 474464	deux 198360	ils 103338	bien 73565
<s> 2494909	il 464543	avec 191192	y 101910	quatre 72649
</s> 2494906	dans 458392	ont 182891	avait 100742	pays 72639
la 1773602	pour 432184	on 173733	tout 100575	lui 71921
l' 1375354	par 385975	mais 171673	nous 99583	président 69009
le 1347102	au 385091	aux 164108	ces 98909	après 68934
à 1237111	cent 368009	sont 159327	fait 95730	faire 65998
les 1142717	mille 345826	cette 147680	même 93869	france 64708
et 1119745	pas 311897	été 144630	être 93540	leurs 60588
des 992380	sur 290381	ou 142066	sans 85229	encore 59245
d' 963222	plus 274100	ses 129876	si 83817	je 58926
en 821863	qu' 263880	comme 129820	était 82843	état 58584
un 755685	neuf 260499	c' 121121	entre 82774	monde 58530
du 740098	monsieur 259947	sa 119048	cinq 82220	autres 57835
717784	ne 245742	</art> 117807	virgule 78989	politique 57271
717784	s' 242904	<art> 117807	où 78137	contre 55976
une 587313	se 236629	pourcent 114234	dont 77588	six 55628
est 525531	n' 235992	trois 114070	aussi 77222	premier 55573
a 509093	ce 229549	leur 114025	depuis 74021	ministre 55111
que 491909	son 198461	elle 108421	ans 73832	très 53714

Figure 5.1: Most frequent words and counts from "Le Monde" 1990-92

the corpus. This assumption might not hold for large corpora, for example because the language might change over the years.

In the end, it is only recognition experiments that can decide whether a LM training corpus is big "enough".

5.2 Text pre-processing

Figure 5.3 shows an example of text from the CD-ROM with "Le Monde" database. The shown text is one story. Every story starts with date and page information, and ends with a block with information about the story. The beginning of this information block is marked by "FICHE DOCUMENTAIRE", then follow a number of fields like subject and the persons appearing in the article. Finally there is an ID that uniquely identifies the story (yymmddLM<number>).

Clearly the information block should be suppressed before the text is passed on to the LM-training program. It would not make sense to train on a sentence like "Noms propres: GORBATCHEV MIKHAIL; MODROW HANS; PARTI COMMUNISTE SOVIETIQUE".

In the present work the information about the story is not used for anything. However, one could imagine training LMs for different subjects and text types. In that case the information block would be useful.

5.2.1 pre1 pre-processing

In figure 5.4 is shown the same text as in figure 5.3 but after the pre1 preprocessing.

The goal of the preprocessing is firstly to filter out "noise" like the leading date information and the tailing information block. Secondly, the text should be turned into a pure text form where all punctuation signs have been removed, and where sentence and paragraph breaks have been detected.

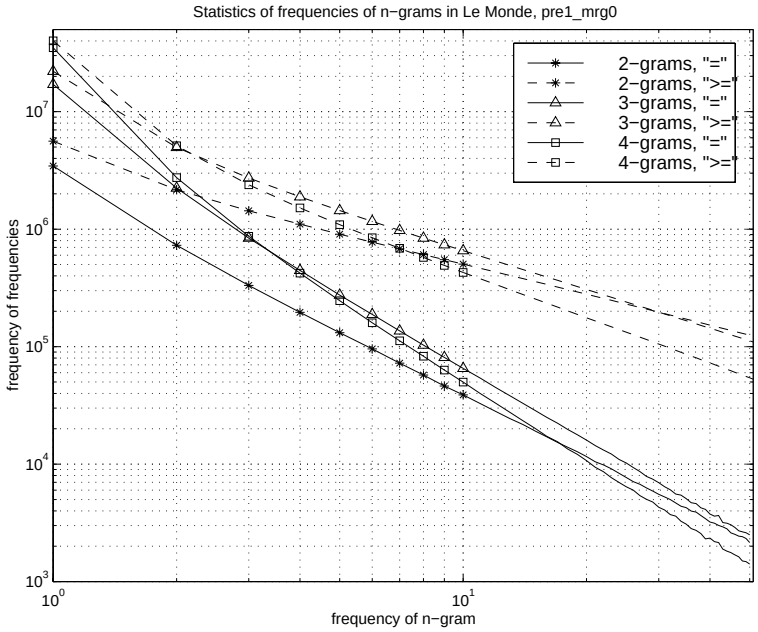


Figure 5.2: Statistics of n-grams in "Le Monde" 1990-92. "=" means how many n-grams occurred exactly 1 time, 2 times, etc. ">=" means how many n-grams occurred at least 1 time, 2 times, etc. Statistics are shown for 2-grams, 3-grams and 4-grams. Data points are only shown up to 10 to make the curves more readable from 10 to 50.

The text form should be the same as if it was read. (e.g. "M" should read "monsieur" and "%" should read "pour cent", etc.).

Much effort has been put into writing and testing the preprocessing script to assure that it is as "clever" as possible. For this purpose, a trial and error approach was adopted, and many temporary versions of the preprocessing were tested before defining the "pre1" version.

Some characteristics of the pre1 pre-processing is:

- abbreviations and special signs are spelled out (e.g. "M" → "monsieur" and "%" → "pour cent", etc.).
- all words are lowercased. (E.g. "URSS" → "urss", "Paris" → "paris")
- numbers are spelled out, and dashes are kept in numbers up to 100. (E.g. 96 → "quatre-vingt-seize")
- dashes in words are kept. (E.g. "Jean-Pierre", "est-ce", "peut-être")
- *but* "t-il", "t-elle" and "t-on" are considered separate words, and are separated from the preceeding vowel. (E.g. "a-t-il" → "a t-il")
- Headlines are skipped. (Sentences ending without a punctuation is considered a headline)
- Number heuristics: "1 000 000" → "1000000" but "12 345 67" remains unchanged.
- split on apostrophe. (E.g. "c'est" → "c' est")

```
Le Monde

31 janvier 1990, page 1

Moscou et Berlin-Est acceptent l'idée de l'unification des deux Etats allemands
M. Gorbatchev veut renforcer ses pouvoirs présidentiels

Recevant à Moscou M. Modrow, le premier ministre est-allemand, M. Gorbatchev
a clairement accepté l'idée de l'unité allemande, à laquelle les dirigeants
de Berlin-Est, dont M. Gysi, président du Parti du socialisme démocratique
(ancien PC), se sont ralliés. D'autre part, M. Gorbatchev a catégoriquement
démenti, mercredi 31 janvier, les rumeurs selon lesquelles il démissionnerait
du secrétariat général du PCUS. La " Pravda " confirme cependant indirectement,
mercredi matin, l'intention de M. Gorbatchev de renforcer ses pouvoirs de
président de l'URSS.

FICHE DOCUMENTAIRE

Sujets - International: ALLEMAGNE; DEMENTI; DEMISSION; INSTANCE; PARTI POLITIQUE;
RDA; RFA; RUMEUR; UNIFICATION; URSS; VOYAGE A L'ETRANGER; VOYAGE ETRANGER
Noms propres: GORBATCHEV MIKHAIL; MODROW HANS; PARTI COMMUNISTE SOVIETIQUE
Taille: BREF

900131LM138598
```

Figure 5.3: Text example from the "Le Monde" database

```
<art>
<p>
<s> recevant à moscou monsieur modrow le premier ministre est-allemand monsieur
gorbatchev a clairement accepté l' idée de l' unité allemande à laquelle les
dirigeants de berlin-est dont monsieur gysi président du parti du socialisme
démocratique ancien pc se sont ralliés </s>
<s> d' autre part monsieur gorbatchev a catégoriquement démenti mercredi
trente_et_un janvier les rumeurs selon lesquelles il démissionnerait du
secrétariat général du pcus </s>
<s> la pravda confirme cependant indirectement mercredi matin l' intention de
monsieur gorbatchev de renforcer ses pouvoirs de président de l' urss </s>
</p>
</art>
```

Figure 5.4: Text example from the "Le Monde" database after pre1 preprocessing (pre1_mrg0).

See the source [1] for a complete description of the details and exception and special cases of the preprocessing program.

[1] see e.g./home/speech/andersen/lm/scripts/fpms/prep_cmu.pl or DAT tapes with backups from FPMs.

5.2.2 pre2 pre-processing

In the previously described pre1 preprocessing some words with dashes were kept together. In pre2 the idea is to let it be totally data-driven what words should be merged. This means that pre2 deletes all dashes in pre1. Afterwards, when a merging of multi-words are performed, the most frequent words will be merged back together.

However, an error was discovered in pre2: "a-t-il" → "a t il" The "t" should have been "t-" to be able to distinguish it from the letter t, because the pronunciations are different: "t": /t e/ but "t-": /t/.

5.3 Multi-word merging

Three multi-word LM preprocessing were performed: with pre1 preprocessing 500 and 1000 multiwords were introduced, and with pre2 processing the case of 500 multi-words was tested.

Figure 5.5 shows the example text after introducing 1000 multi-words (only a few occurs in this text of course).

```
<art>
<p>
<s> recevant à moscou monsieur modrow le_premier_ministre est-allemand monsieur
gorbatchev a clairement accepté l'_idée de_l' unité allemande à_laquelle les
dirigeants de berlin-est dont monsieur gysi président du_parti du socialisme
démocratique ancien pc se_sont ralliés </s>
<s> d'_autre_part monsieur gorbatchev a catégoriquement démenti mercredi
trente_et_un janvier les rumeurs selon lesquelles il démissionnerait du
secrétariat général du pcus </s>
<s> la pravda confirme cependant indirectement mercredi matin l' intention
de_monsieur gorbatchev de renforcer ses pouvoirs de président_de_l' urss </s>
</p>
</art>
```

Figure 5.5: Text example from the "Le Monde" database after pre1 preprocessing and merging of 1000 multi-words (pre1_mv1000).

A number of small scripts and programs were written to automate as much as possible the multi-word preprocessing. [2] The most frequent word tuples (of 2, 3, and 4 words) were selected as multi-words. However, before selecting the multi-words, the count for each tuple was weighted by a function that favors short words:

$$\text{weight}(n\text{-gram}) \;=\; \left(\frac{1}{l_1}+\frac{1}{l_2}+\cdots+\frac{1}{l_n}\right)/n, \tag{5.1}$$

where l_i is the length in number of phonemes of word i in the n-tuple. For words with multiple pronunciation, the average number of phonemes was used, and for OOV words, the number of phonemes was set to twice the number of letters in the word. Typical the number of phonemes will be smaller than that, but this way OOV words are de-emphasized.

The highest weight, 1.0, is done to tuples of one phoneme words. In general, it is the short words in a tuple that has the most influence on the weighting measure.

The reason to introduce the weighting of the word tuples is the observation that French has many short words. By the weighting, the word length will be a bit more conform.

[2] see e.g. the DAT backup from FPMs or scripts in /home/speech/andersen/lm/scripts/fpms/ and /homes/andersen/scripts/dir_lm/

Complementary work could study the effect of other weightings or no weighting at all, only using the frequency of the tuple to determine whether it should be a multi-word. In particular it could be interesting to make a study of which words are most often inserted or deleted, and then favor these words to be part of multi-words.

5.4 Creation of LMs

After the raw text had been transformed into a suitable format, the CMU/Cambridge LM Toolkit was applied.

The toolkit provides several ways to create LMs. One way is to specify a vocabulary at the beginning, and use compact file-formats to store count for word tuples (words are replaced by a number). However, this would mean that each time the vocabulary changes one has to collect statistics again. This can be used to easily generate LMs for several vocabularies, e.g. a 64k and a 20k vocabulary.

Another approach was used, where word tuple counts were stored in ASCII-files where words are spelled out. Each line contains the word tuple followed by the count. Having these statistics files calculated once, it is possible to make LMs with various vocabularies.

Such statistics were calculated for each year separately and then combined afterwards. In this way, it will be much easier later if one wish to create LMs trained on a different amount of text data.

When making the LM, three factors determines the size of the LM: 1) the vocabulary size, 2) the order of the LM, and 3) the cut-offs. In all experiments, the vocabulary size was close to 64k words, varying slightly when multi-words were used, and for pre2 preprocessing. The Toolkit can generate any order or n-gram. In this work bigrams, trigrams, and 4-grams were tested.

The cut-offs, are used to prune the LM. For a trigram, cut-offs=1_4 means that bigrams will only have their own entry in the LM if the word pair occurs more than 1 time in the training data. Similarly the '4' means that that a word triple will only have its own entry in the LM if occurred more than 4 times. For 4-grams, there would be one more cut-off value, applying to the 4-grams, and for bigram LMs, only one cut-off value is needed. The numbers have to be increasing, which assures that if a trigram is in the LM, then the two corresponding bigrams will also have their own entries.

If all cut-offs are 0, there will be no pruning of the LM. However, when trying to create an LM at FPMs with a cut-off of 0, the Toolkit crashed because of memory shortage. It was not discovered whether this problem is due to the program source, hard-ware/OS limitations, or parameters used when the toolkit was compiled. Two computers were used in the LM work at FPMs: hal.fpms.ac.be, and tcts.fpms.ac.be.

The Toolkit also have some more specialized options, that were not tested in this work. See the documentation of the toolkit to get more information.

Chapter 6

MLP training

6.1 The BREF speech database

The French speech database "BREF" was recorded at LIMSI[1] using the recording facility LIMREC. The recordings were made in an acoustically isolated room. The data is sampled at 16 kHz and comprises 5330 utterances of 80 speakers (44 women and 36 men). The sentences were taken verbatim from the French newspaper 'Le Monde' (September/October 1989 and January 1990) and were selected to get a broad coverage of all French phonemes in different contexts.

The speech data used in our experiments were divided into three, non–overlapping sets:

1. **Training set:** 3000 sentences from approx. 56 speakers (approx. 32 female and 24 male);
2. **Cross–validation set:** 737 sentences;
3. **Test set:** 242 sentences.

The training and cross–validation set correspond to a total of 7.5 hours of speech.

6.2 The features and MLPs

The HMM/ANN hybrid system used MLPs with 234 input nodes corresponding to a window of 9 frames, where each frame consists of 12 RASTA-PLP and 12 Δ-RASTA-PLP features as well as the Δ-log- and $\Delta\Delta$-log-energy.

For the context–independent recognition as carried out here, the output layer has 36 units, corresponding to 35 basic phonetic units and a silence state.

As default the MLPs had 1000 hidden units (HU), but some bigger MLPs with 2000 and 4000 HUs were also trained.

6.3 Introduction to the MLP training

In all the following MLP retrainings, the weights for the first iteration (one iteration = one complete MLP training) is initialized with random values. For each epoch (one epoch = one pass through all the training data) the MLP weights are stored in log-files. As opposed to the MLP trained by Dan, these MLP trainings use an iterative scheme, where the weights for one iteration is initialized by the weights stored in the log-file after one epoch of the previous iteration. That is, after one epoch of the "boot-iteration" the weights are stored in a log-file. When "iteration-1" starts, the weights are

[1] LIMSI is the acronym for **L**aboratoire d'**I**nformatique pour la **M**écanique et les **S**ciences de l'**I**ngénieur.

initialized from that log-file. However, for the forced alignment the final MLP (after the last epoch) of the previous iteration is used.

All the MLP training (and forwards passes) was performed on SPERT boards, using fixed point. The MLP training program was the ICSI provided software, qnstrn version v0_96 and qnsfwd v0_96.

6.4 MLP retrain 4: 1000HU, start segmentation from "our-final-mlp", default LR schedule

In MLP retrain 4 the first segmentation file is obtained with "our-final-mlp" (using the boot_weights option) converted from STRUT format to ICSI format. The size of the MLP is 1000 units in the hidden layer.

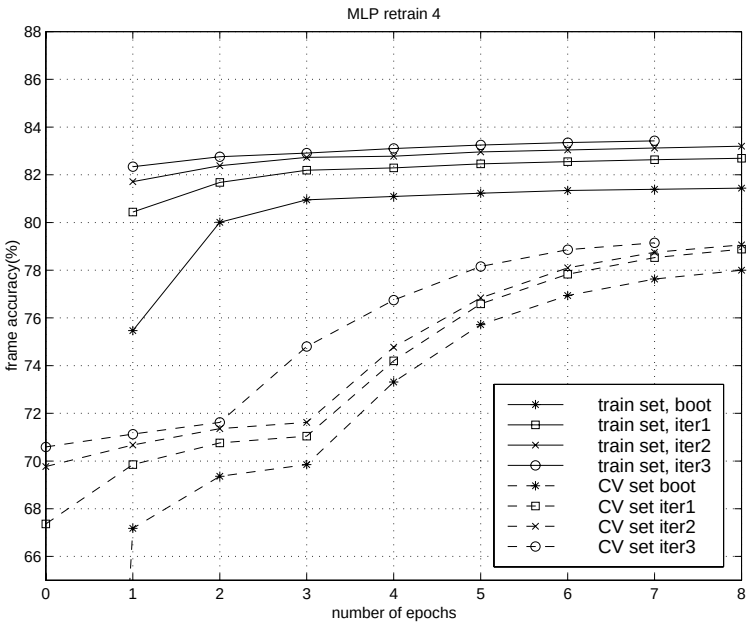


Figure 6.1: MLP retrain 4: frame accuracy

Figure 6.1 shows the frame accuracy on the training set and the cross-validation (CV) set in MLP retrain 4. The frame accuracy is the number of correctly recognized frames. A frame is recognized, if the greatest of the MLP output (taking a max) corresponds to the label in the segmentation file. Note that the segmentation file is different for each iteration.

The learning rate value is 0.08 in the first epochs, and then decays exponentially by dividing it by 2. The exponential decay starts when there is no more improvement on the CV set, or when the improvement becomes too small. In the boot iteration and iteration 1 and 2, there are three epochs with learning rate = 0.08, and it is seen that the frame accuracy on the CV set saturates. An intuitive explanation of this is that the MLP parameters keeps moving around the optimal values without getting the chance to converge. This is because MLP weights are updated for each data frame presented, which makes the method a *randomized* gradient descent method. Then when the learning rate is lowered in the following epochs, it is seen how the frame accuracy improves on the CV set, until the training finally stops, when the improvement gets too small. This is similar to simulated annealing techniques. In all the iterations, the learning rate is 0.00025 in the final epoch.

Epoch 0 denotes the CV performance before the MLP training starts. For the boot iteration, where the weights are initialized with random values, the CV performance is only 3.12 %, and falls out of the graph. For the other iterations, the MLP in epoch 0 is the same as the epoch 1 MLP from the previous iteration. Although it is the same MLP, there is a slight difference in performance because the segmentation files are slightly different.

As one would hope, the frame accuracy reaches larger values for each iteration. It is also seen that performance on the training set is 3-4 % better than on the CV set. With a finite size MLP and an infinite amount of training (and CV) data, ideally the performance should become the same on the CV and training set. Note also, that the frame accuracy for the training set is calculated "on the fly" during the training. This means that for the training set, the numbers reflect the frame accuracy (averaged) *during* the epoch and not *after* the epoch. The CV set on the other hand is tested *after* each epoch.

6.5 MLP retrain 5: 2000HU, start segmentation from "our-final-mlp", default LR schedule

In MLP retrain 5 the first segmentation file is obtained with "our-final-mlp" (using the option "boot_weights") converted from STRUT format to ICSI format. This is the the same as for MLP retrain 4, but here the size of the hidden layer is *2000 units*.

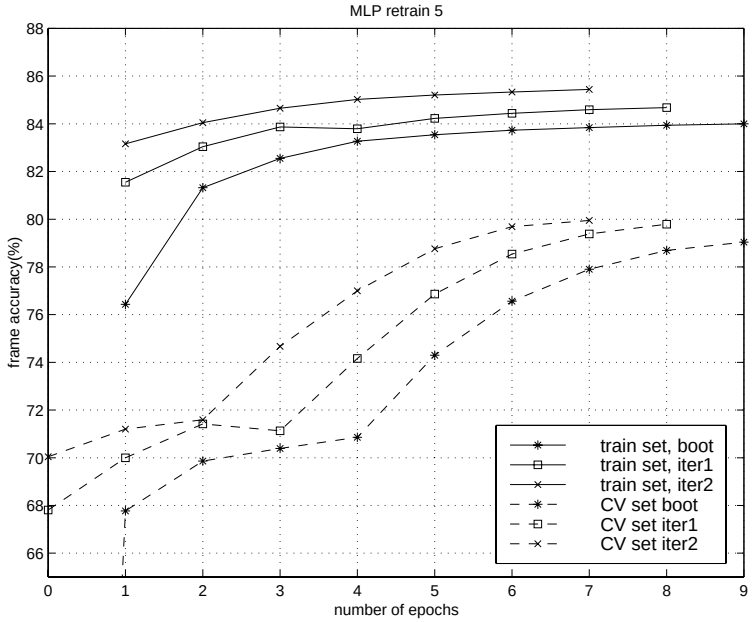


Figure 6.2: MLP retrain 5: frame accuracy

When comparing figure 6.2 to figure 6.1 the same overall tendency is seen: saturation of performance on the CV set after some epochs with fixed learning rate, followed by improvements during the epochs when the learning rate is decaying.

However, if one compares the absolute values one observe that performance is 1-2 % better with the 2000 hidden unit (HU) MLP. And the biggest improvements are observed for the training data. One can interpret this difference as a slight tendency to over-training.

6.6 MLP retrain 6: 1000HU, start segmentation from rule-based system, default LR schedule

In MLP retrain 6 the first segmentation file is not obtained from forced alignment as in MLP retrain 4 and 5. Instead a segmentation file from FPMs is used. This segmentation file was created at FPMs, using a phonetic transcription from a rule-based system which is part of a text-to-speech system (Mbrola).

The size of the MLP is 1000 units in the hidden layer as for MLP retrain 4.

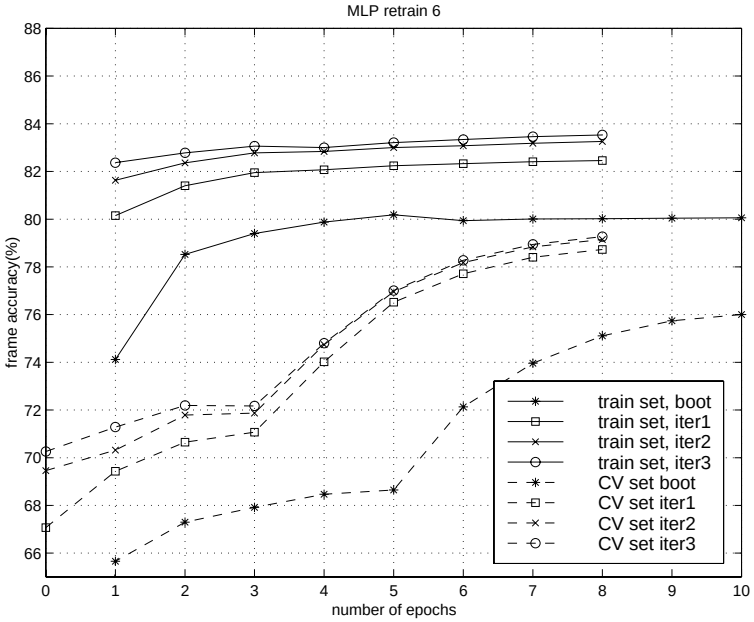


Figure 6.3: MLP retrain 6: frame accuracy

The evolution of frame accuracy for MLP retrain 6 in figure 6.3 is different from retrain 4, in that the performance of the boot iteration is a couple of percent lower while retrain 4 and retrain 5 reaches almost the same accuracy in the following iterations. The lower performance during the boot iteration indicates that the forced alignment from FPMs is harder to learn and/or doesn't match the acoustic as well the following re-segmentations.

The idea behind this MLP-training is that the MLP learns a bit from the first segmentation file from FPMs, and then it continues to learn from the new segmentation files in iteration 1, 2 and 3. One might think that such an MLP "gets a broader education". Or in other words, maybe one segmentation is not always correct, so training on several segmentations, the MLP might learn some from each, and maybe generalize better. It might be somewhat similar to what happens when soft-targets are used, that is probability vectors instead of just zero's and one's.

6.7 MLP retrain 6b: 1000HU, start segmentation from rule-based system, fixed LR schedule

MLP retrain 6b uses the same training scheme as MLP retrain 6 but with fixed learning rate schedule, with a slower decrease. The learning rate still starts with 0.008 for some epochs, followed by an

exponential decrease of the learning rate. In the ICSI soft as well as in STRUT, the default learning rate schedule is the "New-bob", which uses a decrease factor 0.5, while it is only $\sqrt{0.5}$ in retrain 6b.

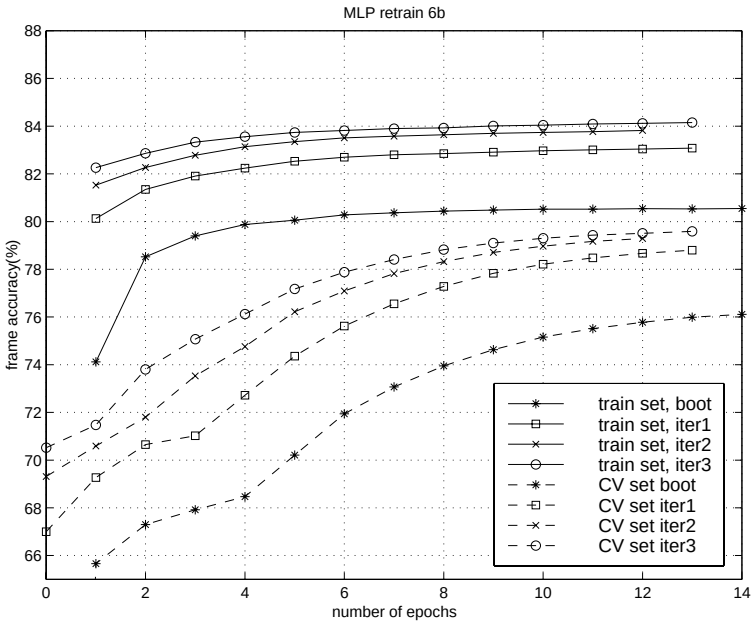


Figure 6.4: MLP retrain 6b: frame accuracy

In the boot-iteration, there are 4 epochs with fixed learning rate of 0.008, and in the following iterations, there is one less 0.008 epoch for each iteration. It is seen that when the learning rate starts to decrease (in epoch 5, 4, 3, and 2, resp.) the per epoch improvements are smaller than in MLP retrain 6, because the learning rate decrease is smaller than in retrain 6. However the MLP reaches slightly higher maximum values, even though it uses more epochs to get there.

The reason for this experiment was the question: do we *force* the MLP to converge by the exponential decay of learning rate? This experiment shows that a slower decay doesn't change much compared to the default "New-bob" schedule.

6.8 MLP retrain 7: 1000HU, start segmentation from "Dan MLP", default LR schedule

In MLP retrain 7 the first segmentation file is obtained with the MLP trained by Dan ("First_Run") which is better than "our-final-mlp". However, for historical reasons, the "our-final-mlp" was used in retrain 4 and 5.

In this MLP training, the hidden layer has 1000 hidden units, so MLP retrain 7 only differs from MLP retrain 4 in that the first segmentation (for the boot-iteration) supposedly is better.

In terms of frame accuracy the performance in retrain 7 is very similar to the three first iterations of retrain 4.

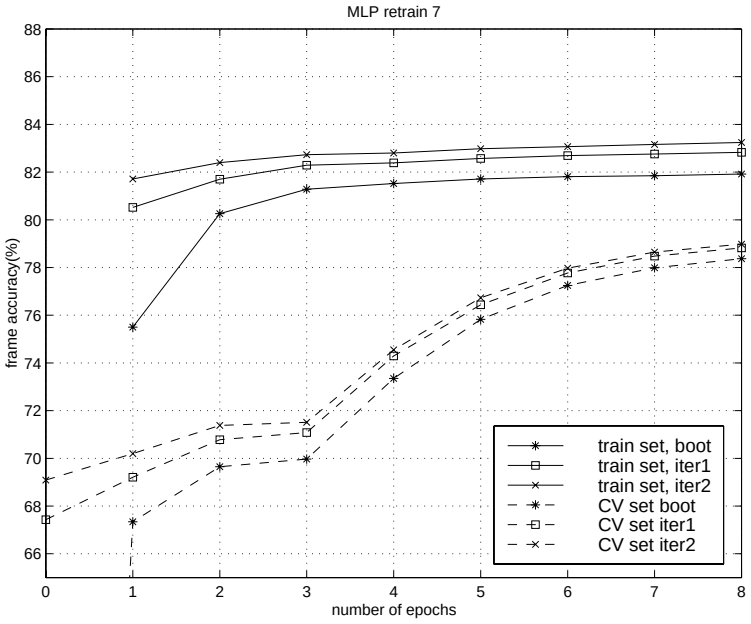


Figure 6.5: MLP retrain 7: frame accuracy

6.9 MLP retrain 8: 2000HU, start segmentation from rule-based system, default LR schedule

MLP retrain 8 only differs from MLP retrain 6 by having 2000 hidden units instead of 1000. As in retrain 6 the FPMs segmentation is used in the boot-iteration.

As in retrain 6, figure 6.6 shows that the boot iteration has a distinct lower performance than the following iterations. But as a result of the larger MLP, the frame accuracy is up to 1 % better on the CV set and 2-3 % on the train set.

The fact that improvements are bigger on the train set indicates a tendency to over-training.

6.10 MLP retrain 9: 4000HU, start segmentation from rule-based system, default LR schedule

MLP retrain 9 is like MLP retrain 6 and MLP retrain 8, but the size of the MLP is increased to 4000 units in the hidden layer.

The comments from retrain 8 also apply to retrain 9 except that frame accuracies are improved by yet another 0.25-2 % on the train set and 0.5 % on the CV set. For the train set, the best frame accuracy in retrain 9 is obtained after the last epoch of iteration 1, and is 87.01 % and for the CV set 80.55 % is obtained after the last epoch of iteration 2. The corresponding best numbers from retrain 6 (using 1000 HU) were 83.53 % and 79.27 %. So the difference in train and CV performance has doubled from 3.26 % to 6.46 %. In other words, the CV performance has only improved 1.28 % while the train set performance improved 3.48 %. And again, this indicates an over-training: an MLP that might be too big, compared to the size of the training set.

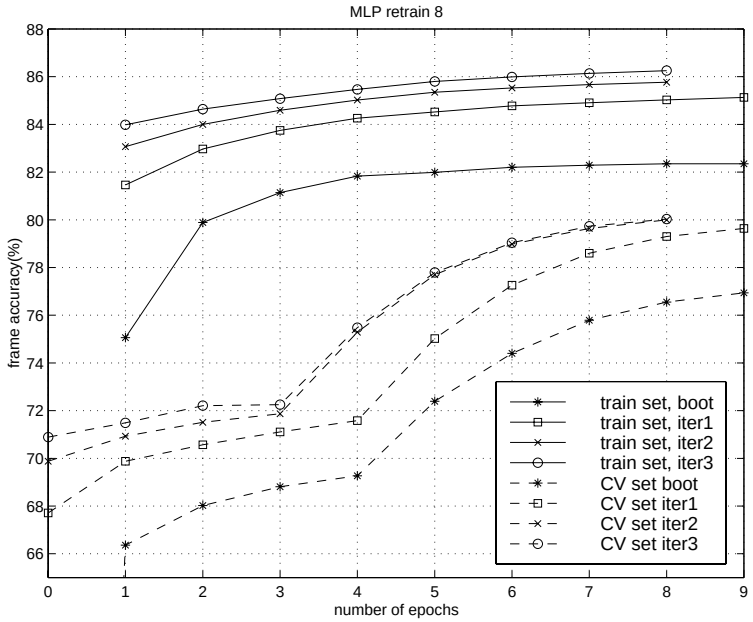


Figure 6.6: MLP retrain 8: frame accuracy

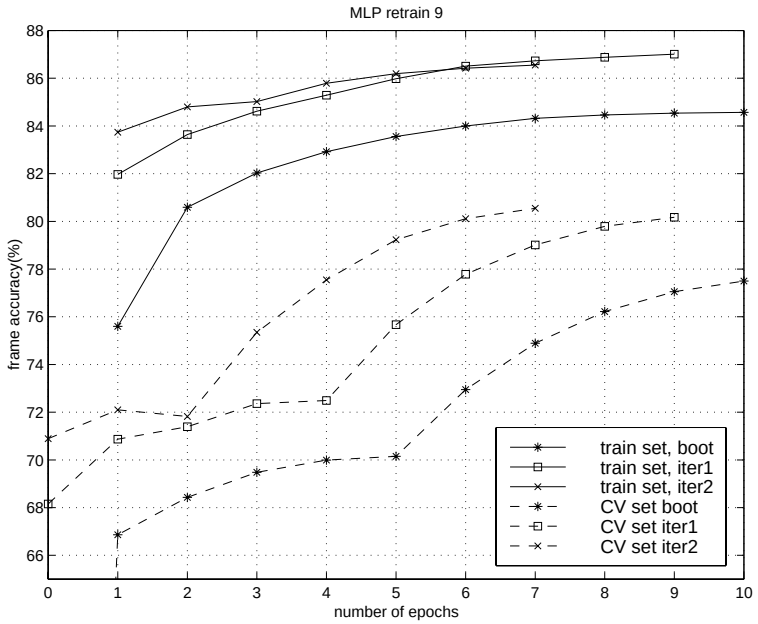


Figure 6.7: MLP retrain 9: frame accuracy

6.11 MLP retrain 10: 1000HU, start segmentation from retrain6b-boot, fixed LR schedule

MLP retrain 10 is similar to retrain 4 and retrain 7 in that it uses 1000 hidden units, and creates the target labels for the boot iteration by a forced alignment. The MLP used for the forced alignment

is that from the boot iteration of retrain 6b. But as opposed to retrain 6b, the MLP training using this segmentation is started from random initialization. In other words: the segmentation for the boot iteration of retrain 10 is the same as was used in iteration 1 of retrain 6b. The fixed learning rate schedule from retrain 6b is used, with two more iteration added, so that the learning rate ends at 0.000125.

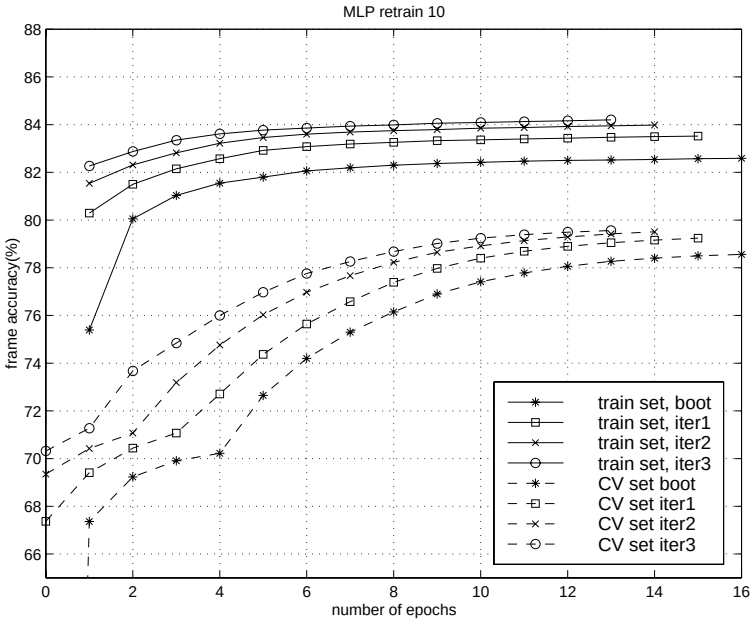


Figure 6.8: MLP retrain 10: frame accuracy

The effect of the new learning rate schedule is very smooth curves for the frame accuracy. They are similar to the last three iterations of retrain 6b. When comparing to retrain 7, the smaller changes in learning rate are obvious, and it is also observed that the final frame accuracies are higher.

6.12 MLP retrain 11: 1000HU, start segmentation from "Dan MLP", fixed LR schedule

MLP retrain 11 repeats retrain 7, but with the same fixed learning rate schedule as retrain 10. That is: the first segmentation file is obtained with the MLP trained by Dan ("First_Run").

Except for the first epochs, the curves for retrain 11 is almost identical to retrain 10. The difference in the beginning of each iteration reflects the fact that the first segmentation in retrain 10 and retrain 11 are different, and that iter1 is initialized with the MLP from boot, epoch 1, and so forth.

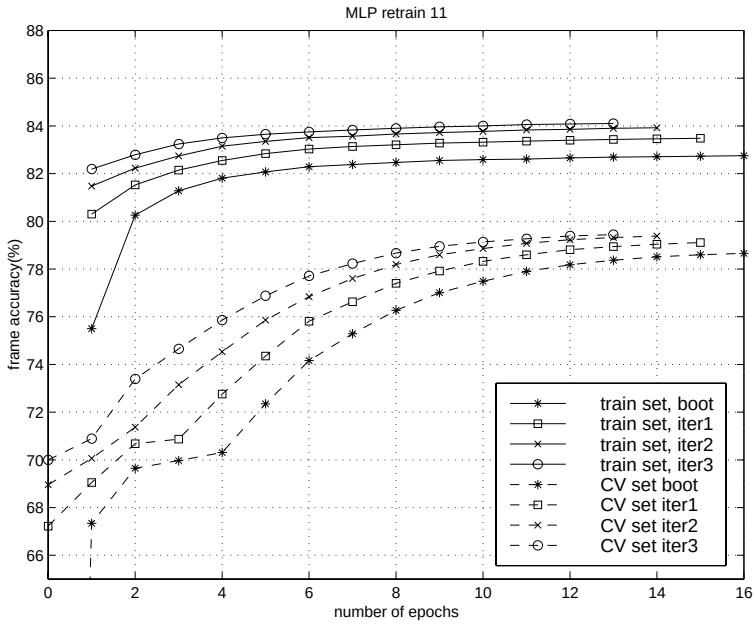


Figure 6.9: MLP retrain 11: frame accuracy

Chapter 7

Speech recognition experiments

In this chapter all the speech recognition experiments are presented. Although LM training, MLP training and the recognition experiments performed in parallel in an interactive process, all the results will be presented here. Hopefully this will be more readable than if training and recognition were reported in chronological order.

7.1 Comparing experiments

7.1.1 Execution time of recognition experiments

It is difficult to give exact figures of the execution times, because the different experiments were run under slightly different conditions. Most recognition experiments were run on one of the following types of computer, all running at 300 MHz:

- a 4 processor Sun ULTRA Enterprise 450 with 1 GB of memory + 1 GB swap disk (bishorn), or
- a single processor Sun SPARC Ultra 30 with 384 MB memory + swap (ruinette, luisin), or
- a single processor Sun SPARC Ultra 10 with 256 MB memory (in most cases) + swap (arolla, blanc, cry, dom).

It was avoided to run "noway" processes that did not (almost) fit into memory because the program can start to run *very* slow if it starts to swap. For some sentences, that are hard to decode, the search space can grow big, and noway seems to need random access to all parts of the allocated memory, which means that the whole process should be in loaded in memory to avoid the "swap death". On easier parts of the recognition, the search space will be smaller, and some of the allocated memory might not be used, and thus can be swapped out without causing any trouble.

In addition to different hardware, the run-time conditions also vary in terms of different load. However, if possible it was avoided to have a load of more than 200 % per processor, and in many experiments the decoding process gets almost 100 % processor time. Because of the different conditions the execution times can be seen as worst case numbers that provides some guidelines for execution times.

To get comparable figures, all experiments should have been run on the same machine with no other "competing" processes. It has not been possible to reserve a machine to these experiments however, but is a possibility in the case where exact numbers are required (e.g. to test the exact effect of various settings of pruning parameters). Looking back, one could suggest to use the unix command "time" to measure the execution time in terms of CPU time. However, this measure might not be completely invariant to the load of the machine either.

7.2 Significance-level of results

Figure 7.1 shows significance intervals for WERs on the test set. The intervals are very conservative estimates, because all errors are assumed independent. That is, it is assumed, that for each word, the probability p for an error is the same, and independent of whether the words around were correct or not. Then the probability for observing n errors is

$$P(\#\text{error} = n) = \left(\begin{array}{c} n \\ N \end{array} \right) p^n (1-p)^{N-n}, \tag{7.1}$$

where N is the number of words in the test. The observed WER, $\hat{p}$ is

$$\hat{p} = \frac{\#\text{error}}{N} = E\{p\}. \tag{7.2}$$

In figure 7.1, the x-axis shows the "true" (unknown) WER, p, and the y-axis shows the distribution of the observed/measured WER, $\hat{p}$, represented by the mean value of the distribution (7.1) as well as the 90% interval and the 99%interval.

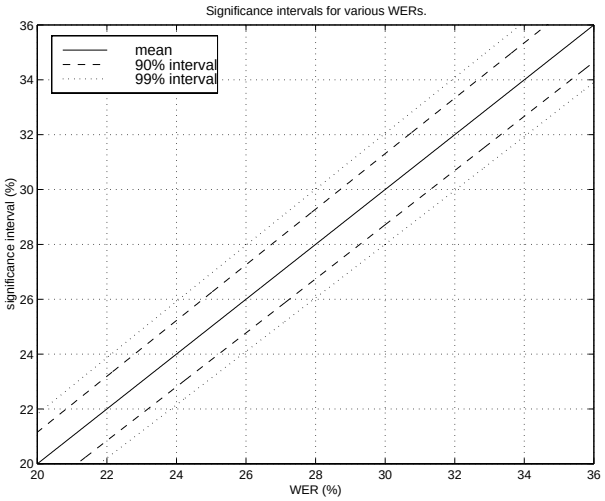


Figure 7.1: Significance intervals for different values of WER. The 90% interval and 99% interval are shown.

It can be seen, that for the given test set, and with the above assumption, the observed WER will be within ±2% from the "true" WER with 99% certainty. And within ±0.6% with 90% certainty.

However, when comparing two recognition experiments, the results does not need to be 4% apart to say that one method is better than the other. Although it has not been done in this report, it is possible to make more strict statistical tests. If it is counted how many errors are the same by two recognition methods, and how many errors are different, one can make test, where the overall WER need not be far apart to tell that one method is significantly better than the other. Many of the recognition errors are likely to be the same in different recognition experiments.

7.3 History of recognition experiments

The history of this chapter is the story of an on-going development where many different parameters have changed along the way. Some mistakes in early experiments were corrected in later experiments, which means that some of the experiments are difficult to compare.

Before presenting all the results, a short outline of the experiments:

Firstly, in the early experiments, the "Dan MLP" was used, before new MLPs were trained. But the "Dan MLP" was also used in later experiments, using this MLP as a kind of baseline MLP to compare the effects of different LMs.

Three typical sizes/types of LMs were (in order of increasing complexity): bigrams with cut-offs=3, trigrams with cut-offs=2 (the default/baseline), and 4-grams with cut-offs=1.

The first experiments used LMs trained on all "Le Monde" 1990-92. Later when it was discovered that some BREF sentences were in January 1990, it was attempted to filter out these sentences. However, because of different processing and formatting of the BREF transcripts and the "Le Monde" data, it was impossible to say if all BREF sentences had been filtered out from the LM training data. So it was decided to leave out all January 1990.

The three versions of LM train set was given the names "All January 1990" (the original default), "Some January 1990" ("no_bref" was added to the file-names because it was believed to contain no BREF data), and "No January 1990" ("no_bref_month" was added to the file-names). In the later experiments "No Jan'90" is the default.

The testing of the two LM text preprocessing methods, "pre1" and "pre2" as well as the experiments with multi-word merging used "All Jan'90" and the Dan MLP.

"Some Jan'90" was used in MLP retrain 4, 5, 6, 6b, and 7. These experiments used the default size LM: a trigram with all cut-offs=2. However, some experiments with the "Dan MLP" used other LMs to get comparisons to the "All Jan'90" LMs.

From MLP retrain 8 and on, the "No Jan'90" LMs were used, and retrain 6b and 7 MLPs were also run with "No Jan'90" LMs to allow comparisons with later MLP retrain 8 and on, as well as to monitor the influence of BREF data in the LM training set.

The influence of acoustic scaling in the decoding, was tested with the "Dan MLP" on "Some Jan'90" and "No Jan'90", and with MLPs from retain 8, 10, and 11 on "No Jan'90".

7.4 "Dan MLP", "All Jan'90", LM preprocessing and LM size

All the experiments in this section use the "Dan MLP".

Before the LM work presented here, IDIAP had French LMs that had been created by FPMs, using the same training data: All months of "Le Monde" 1990-92. However, the preprocessing was more simple and thus might have caused some bugs in the text that were fed to the LM training program.

"Dan MLP", FPMs LMs				
LM	cut-offs	# params	# err	WER
3gr.apr16.98	3 3	?	1104	32.1 %
3gr.aug13.98	3 3	?	977	28.3 %

Table 7.1: Two LMs created by FPMs, and tested at IDIAP.

Table 7.1 shows the performance of two FPMs LMs using the "Dan MLP". The dates indicates when the LMs were down-loaded from FPMs. There are not much information about these LMs, except that they were trained on 1990-92 of "Le Monde" (="All Jan'90"), and cut-offs=3 were used. This means that we don't know why one performs so much better than the other.

7.4.1 pre1 preprocessing, no multi-words

This section describes results obtained with LMs that used the pre1 preprocessing. "mrg0" means that no multi-words were used (no merging). Remember that these results are too optimistic because there are BREF data in part of the LM training set (January 1990).

pre1_mrg0				
n	cut-offs	# params	# err	WER
3	4 4	2,412,924	967	28.0 %
3	3 3	3,055,152	945	27.3 %
3	2 2	4,228,346	909	26.3 %
3	1 1	7,195,167	855	24.7 %
4	2 2 2	6,615,364	886	25.6 %
4	1 1 1	12,327,963	812	23.5 %

Table 7.2: Results with different LMs based on the pre1 preprocessing, using no multi-words.

Comparing the trigram with cut-offs=3 in table 7.2 and the best trigram in table 7.1, one sees that the new preprocessing gains 1% (abs) in WER.

Table 7.2 also shows how performance increase with the size of the LM. However, as will be shown later, this increase is less pronounced when there is no BREF data in the LM training set.

7.4.2 pruning parameters

Table 7.3 shows experiments with various setting of decoding parameters (pruning), all using the default type LM: a trigram with all cut-offs=2, and the default MLP: the "Dan MLP".

pre1_mrg0, 3gr_2_2, diff. pruning								
name	n.h.	prob_min	b.	s.b.	× real-time	# err	WER	WER (part.)
default	7	0.00025	4	5	39	909	26.3 %	25.7 % / 73 sent
run2	5	0.0025	3	4	5.0	1058	30.6 %	
run3	14				74	879	25.4 %	
run4		0.000025			344	-		24.9 % / 73 sent
run5			6	7	79	909	26.3 %	
run6	14	0.000025	6	7	Killed! (13 sent. in 17 hrs. + memory problem...)			

Table 7.3: WER for different pruning values. 'n.h.' is n_hyps, and 'b' is beam, and 's.b.' is state_beam. For run2-6 values are only shown when different from the default.

The default parameter setting of Noway was:

- -n_hyps 7
- -prob_min 0.00025
- -beam 4
- -state_beam 5
- -acoustic_scale 0.15
- -new_lub

Run2 in table 7.3 differs from the default by n_hyps=5, prob_min=0.0025, beam=3, and state_beam=4. This increased pruning speeded up the decoding by a factor 8, from 39 times real-time to 5.0 times real-time, but achieved this at the cost of 4.3% higher WER (abs).

The experiments that had less pruning than the default (run3-6) all run slower than the default, and some obtain a better result. Run3 and run5 run at about the same speed, but only run3 improves the default result (0.9% abs). So increasing n_hyp is better than increasing the beam-width (in this

case). Lowering prob_min by a factor 10 (run4) also improves WER with about 1% (abs), but was running so slow, that it was killed before terminating.

These experiments with different pruning parameters show some of their impact. However, it could interesting to do a more extensive study to determine the optimal compromise for, say, a real-time system and a 10× real-time system.

Though it should be remembered, that the optimal parameter setting most likely is task dependent. And also, the decoding time for a given parameter setting will vary with different tasks. Clean and clearly pronounced speech decode much faster than noisy speech.

7.4.3 pre1 preprocessing, 500 and 1000 multi-words

pre1_mw500				
n	cut-offs	# params	# err	WER
2	2	1,893,013	1014	29.3 %
3	4 4	2,395,710	958	27.7 %
3	2 4	3,106,450	942	27.3 %
3	2 2	4,409,986	886	25.6 %
4	2 4 6	3,509,211	934	27.0 %
4	2 2 2	5,976,468	873	25.3 %

Table 7.4: Results with various LMs using pre1 preprocessing and 500 multi-words.

Table 7.4 shows results when the vocabulary is augmented with 500 multi-words. The same pattern is observed again: more LM parameters gives better performance.

Table 7.5 deviate from this tendency as it is the trigram that performs best. But the 4-gram is still better than the bigram.

pre1_mw1000				
n	cut-offs	# params	# err	WER
2	2	1,984,040	1002	29.0 %
3	2 2	4,367,744	887	25.7 %
4	2 2 2	5,765,542	895	25.9 %

Table 7.5: Results with three different LMs using pre1 preprocessing and 1000 multi-words.

In all the multi-word experiments, the pronunciation dictionary for the multi-words were obtained as a simple combination of the implied words. This implies that for some of the 4-tuples many possible pronunciations are generated. If each word have e.g. 2 pronunciations, the 4-tuple will have $2^4 = 16$ pronunciations, and if each word have e.g. 4 pronunciations, the 4-tuple will have $4^4 = 256$ pronunciations. The prior probabilities of the pronunciations for each word is simply multiplied together to get the prior probability for each pronunciation for the multi-word. That is, we assume that the pronunciations of the composing words are independent (which we know is not true).

This combination of pronunciations implies that the effect of introducing multi-words is only on the LM level.

It is likely that lexicon training on the multi-words will improve recognition. A lexicon training is expected to capture liaisons and other cross-word phenomena within the multi-words. This will make the multi-words more discriminant, and thereby should improve recognition.

7.4.4 pre2 preprocessing, 0 or 500 multi-words

Table 7.6 shows results with pre2 preprocessing and using 0 or 500 multi-words.

pre2					
merge	n	cut-offs	# params	# err	WER
mrg0	3	2 2	4,234,070	940	27.2 %
mw500	2	2	1,857,579	1030	29.8 %
mw500	3	2 2	4,420,153	902	26.1 %
mw500	4	2 2 2	6,117,622	901	26.1 %

Table 7.6: Results with various LMs using pre2 preprocessing and 0 or 500 multi-words.

7.4.5 Comparing pre1/pre2 processing and 0/500/1000 multi-words

The vocabulary is slightly different in the different cases, which makes it difficult to make strict comparisons. Pre1 LMs without multi-words use a vocabulary that was extracted from a 64k word LM from FPMs, whereas the multi-word LMs and pre2 LMs use the (approximately 64k) most frequent words in the LM training data.

Figure 7.2 shows performance in WER as a function of the number of parameters in the LM. With a logarithmic x-axis, the points are seen to approximately follow a linearly decreasing function, when looking at all the points. However, as shown later, when January 1990 is left out of the LM training set, this linear function becomes more flat.

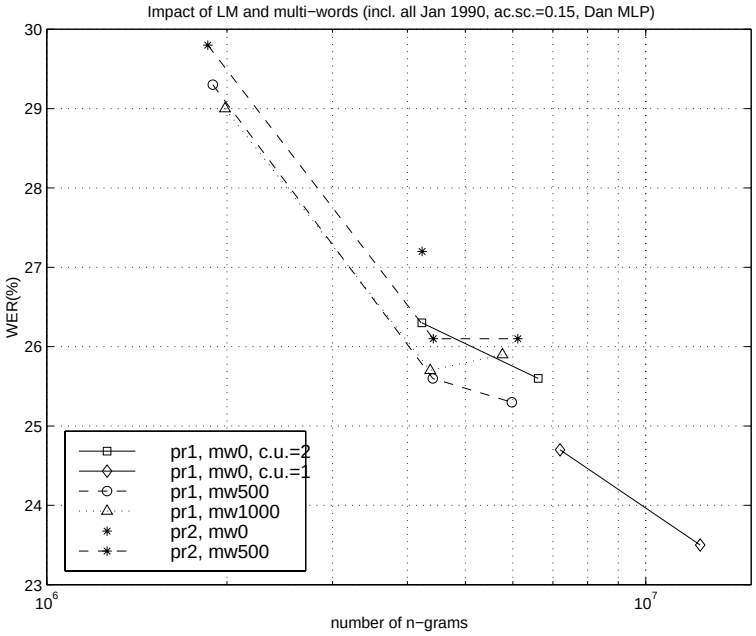


Figure 7.2: Comparison of pre1 and pre2 preprocessing, and 0 or more multi-words. All LMs use "all Jan'90" for LM training, and the LM was the "Dan MLP".

Figure 7.2 compares the two preprocessing methods, pre1 and pre2, and shows the influence of multi-words. All the LMs used cut-offs=2, except for the two marked with diamonds, which used cut-offs=1 (trigram and 4-gram). Bigrams, trigrams, and 4-grams are plotted (if results are available), and are situated around 2M, 4.5M, and 6M parameters.

All LMs using pre2 is marked with stars, and the rest uses pre1 preprocessing. With no multi-words, pre2 (single star) has 0.9% higher WER than the corresponding trigram using pre1 (left square). And

comparing the two dashed lines, one sees, that pre2 also is worse (0.5-0.8%) when using 500 multi-words. The explanation of the lower performance of pre2 could be due to the bug that were discovered later: pre2, which skips all "_" and "-", accidentally also skips "-" in e.g. "t- il" which is an error because it is then no later possible to distinguish "t" and "t-", which have different pronunciations (/te/ and /t/ respectively).

Using 500 multi-words consistently performs better than the corresponding LMs without multi-words. Multi-words win with about 0.5% for pre1, and more than 1% for pre2.

Using 1000 multi-words is better than no multi-words for bigrams and trigrams, but worse for 4-grams. In general, going from trigram to 4-gram improves more in the case of no multi-words (the two squares) than when multi-words are used. This might indicate that using a trigram with multi-words somewhat simulates the effect of having a 4-gram (or even a 5-gram).

Finally, figure 7.2 shows the gain by using cut-offs=1 (diamonds) in stead of cut-offs=2 (squares) in the case of pre1 preprocessing without multi-words. Of cause the price paid for this improvement is an increase in the number of parameters. A possible future experiment would be to see if multi-word LMs using cut-offs=1 would be even better.

7.5 "Dan MLP", "Some Jan'90", LM size and acoustic scaling

"Some Jan'90" indicates that there are probably some of the BREF sentences (partly selected from January 1990 of "Le Monde") in the LM training set. "Some Jan'90" is also called "no_bref" (especially in the filenames) because it was first believed to contain no BREF data. It was attempted to filter out BREF sentences from January 1990, but it is likely that only some of the BREF sentences got caught by this filter.

All the experiments with "Some Jan'90" uses pre1 preprocessing, and no multi-words.

pre1_mrg0				
n	cut-offs	# params	# err	WER
3	3 3	3,053,206	989	28.6 %
3	2 2	4,225,668	995	28.8 %
4	2 2 2	6,610,837	981	28.4 %
4	1 1 1	12,319,472	928	26.9 %

Table 7.7: Results using 4 different LMs trained on "Some Jan'90" using pre1 preprocessing, and no multi-words

Table 7.7 shows the impact of the size of the LM. Again the tendency is that performance increases with the number of parameters. But the increase is less than when "All Jan'90" was used. See a later graphical comparison of "All Jan'90", "Some Jan'90", and "No Jan'90".

pre1_mrg0,		
acoustic_scale	# err	WER
0.15 (default)	928	26.9 %
0.18	923	26.7 %
0.21	927	26.8 %
0.24	943	27.3 %

Table 7.8: Results with different values of acoustic_scale, using a 4-gram with cut-offs=1, trained on "Some Jan'90". The MLP is still the "Dan MLP".

Table 7.8 shows the influence of different values of acoustic_scale using a 4-gram. In a later graph it will be compared to other LMs.

pre1_mrg0 no_bref, diff. settings			
change from default	ac.sc.	# err	WER
	0.15 (def.)	995	28.8 %
	0.18	964	27.9 %
	0.21	967	28.0 %
	0.24	964	27.9 %
	0.27	970	28.1 %
	0.30	987	28.6 %
flt-point (Sun)	0.15 (def.)	999	28.9 %
full forward	0.15 (def.)	1098	31.8 %
cross sentence	0.15 (def.)	995	28.8 %

Table 7.9: Results with different values of acoustic_scale and other recognition parameters, using the default type LM (3-gram with cut-offs=2), trained on "Some Jan'90". The MLP is still the "Dan MLP".

Table 7.9 shows the influence of acoustic scaling. A later graph will compare these results to other LMs.

The table also shows a slight decrease in WER when a floating point version of MLP-forward program is used (run on a Sun workstation) as opposed to the default fixed-point version of the MLP program that run on the SPERT board.

The full forward (full likelihood/full posterior) decoding is obtained by setting `-forward_process` and `-merge_hyps` in noway. This is opposed to the default which is "best path" decoding also called Viterbi decoding. It is not sure why the full forward decoding is 3% worse than the best-path decoding. One explanation could be that the MLP is trained by a best-path method, and thus works best with a best-path decoding. However, it can not be ruled out that other settings of acoustic scaling and language-model scaling are needed to make the full forward decoding work.

The cross sentence decoding option gave exactly the same output as the default. This option was tested because the notion of a "sentence" in the LM is likely not always to be the same as in BREF. That is, during the LM preprocessing, markers are inserted at beginning ("<s>") and end ("</s>") of sentences. At the time BREF was designed they might have used different rules to determine beginning and end of sentences. Using cross sentence decoding should allow more than one "LM-sentence" per acoustic utterance.

7.6 "Dan MLP", "No Jan'90", pre1 preprocessing

In the following the LMs use "Le Monde" 1990-92, except all January 1990 which was left out to completely avoid any BREF sentences in the LM training set. All LMs trained on "No Jan'90" use pre1 preprocessing.

The default MLP is still the "Dan MLP", but new MLPs will also be tested. The default LM type is a trigram with all cut-offs=2, which have 4,126,361 parameters.

pre1_mrg0, no_bref_month				
n	cut-offs	# params	# err	WER
2	3	1,146,276	1146	33.2 %
3	2 2	4,126,361	1005	29.1 %
4	1 1 1	12,004,745	966	28.0 %

Table 7.10: Results with the 3 most used LMs, using default pruning and the "Dan MLP".

With the 3-gram-2-2 the same result was obtained with Guilia's Noway version with floating point LM scale, although it seemed to run slower. It might be better to have a LM scale slightly lower than 1.0 to compensate for the approximation committed when assuming n-order Markov properties for the LM.

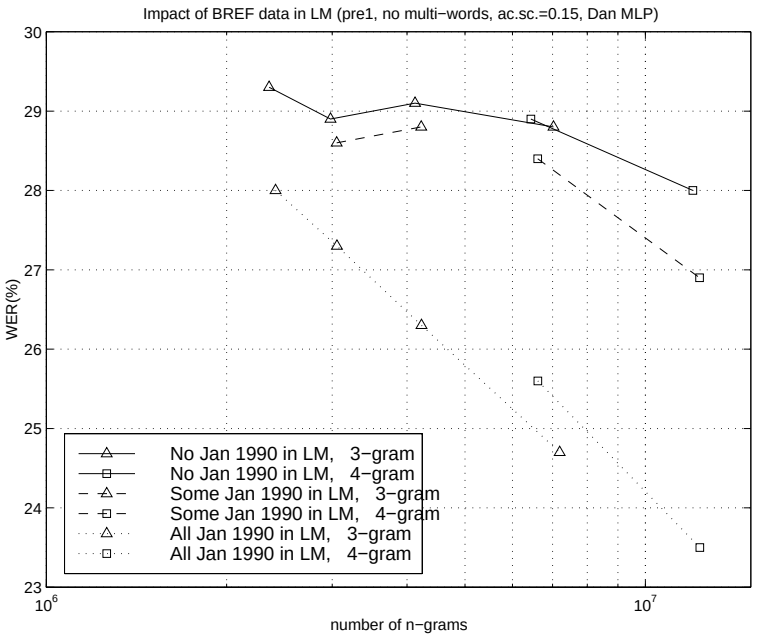


Figure 7.3: Summary: Effect of test data in LM train text. (Dan MLP, "all/some/no Jan'90", 3/4-grams)

Before proceeding with the results using "No Jan'90", figure 7.3 shows a comparison between "All Jan'90", "Some Jan'90", and "No Jan'90". For "No Jan'90", 4 trigrams are shown (triangles, solid line), with all cut-offs equal to 4, 3, 2, and 1 respectively from left to right, and 2 4-grams with all cut-offs equal to 2 and 1.

For "All Jan'90" results are shown for the same type of LMs, and for "Some Jan'90" are shown for some of the corresponding LMs. It is seen that for "No Jan'90" the points can be approximated by a slowly decreasing function of about 1.5-2% (abs) decrease in WER for by using 10 times as many parameters. (one can not assume that this tendency can be extrapolated)

When looking at the points corresponding to "Some Jan'90" and "All Jan'90" it is clearly seen, that the more BREF data there are in the LM training set, the steeper becomes this function. If one had had a LM training set with all the BREF sentences, one would expect the curve to get very close to 0% WER if only the LM had enough parameters, because the LM would "learn" the BREF sentences then.

However, the non-cheating LM training set "No Jan'90" gives little hope to obtain large improvements simply by increasing the number of parameters (at least for this size of training set).

In figure 7.4 results with "No Jan'90" (solid lines) and "Some Jan'90" (dashed line) are compared for different values of acoustic scaling. Results are shown for a trigram with cut-offs=2 (triangles) and a 4-gram with cut-offs=1 (squares).

As could be expected, two general tendencies are observed: 4-grams perform better than trigrams, and "Some Jan'90" (with some BREF data present) performs better than "No Jan'90". What concerns the shape of the curves, the curves for "Some Jan'90" are more smooth than for "No Jan'90". There

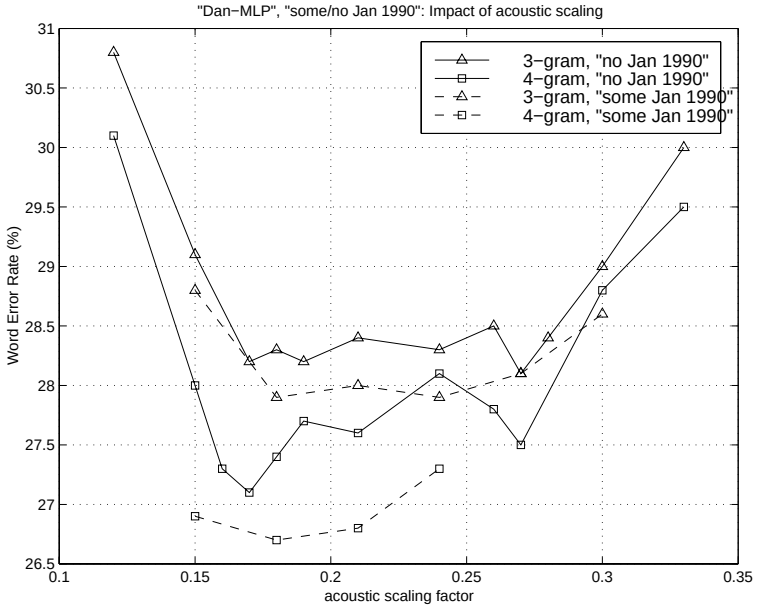


Figure 7.4: Impact of acoustic scaling and having test data in LM train set. (Dan MLP)

seems to be no obvious explanation for this behavior.

At a rough scale, the curves look similar with a large U-shape. However, because of some irregularities the curve for "No Jan'90" 4-grams is almost W-shaped. It can also be observed that the two "No Jan'90" curves looks alike, and the two "Some Jan'90" curves look alike.

A more typical example of similarity between performance curves will be shown in the later figure 7.9, which compares different MLPs and different LMs, but all based on "No Jan'90".

Even though the two curves for "No Jan'90" looks similar, the difference between the trigram and the 4-gram varies from 0.2% to 1.1%. In other words, when comparing these two LMs, the conclusion depends very much on the chosen acoustic scaling factor.

For "No Jan'90" the fluctuations means that for an acoustic scaling factor between 0.17 and 0.27 the WER varies between 28.1-28.5% (0..4%) for the trigram and between 27.1-28.1% (1%) for the 4-gram. And for acoustic scaling factors in the range 0.15-0.30, the performance variation becomes 1% for the trigram and 1.7% for the 4-gram. These differences are quite large compared to the differences due to different LMs.

As "Some Jan'90" is expected to better represent the test data than "No Jan'90", one would expect that relatively less emphasis should be put on the acoustic model when using "Some Jan'90". In other words, a priori one would expect that the acoustic scaling that yields the minimum WER would be smaller for "Some Jan'90" than for "No Jan'90". Due to the large fluctuations for "No Jan'90", it is hard to confirm or reject this a priori expectation.

Figure 7.5 shows a summary of all LMs trained on "No Jan'90" using the "Dan MLP", and the default decoding parameters (e.g. with the acoustic scaling $= 0.15$). In the upper left corner the bigram results are grouped with cut-offs equal to 9, 3, 2, and 1 respectively from left to right.

The gap between bigrams and the higher order LMs is quite remarkable, and even when the acoustic scale is optimized for the bigram (see figure 7.9), the bigrams still lacks about 4% after trigrams and 4-grams.

A first glance on the WER for the different trigrams and 4-grams, tells that they approximately follow a slowly decreasing function. To better distinguish the different trigrams and 4-grams, figure 7.6

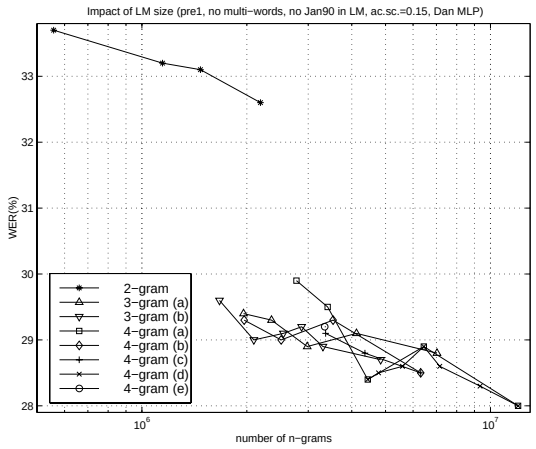


Figure 7.5: Summary of LM impact. (Dan MLP, "no Jan'90", 2/3/4-grams)

zooms in on those.

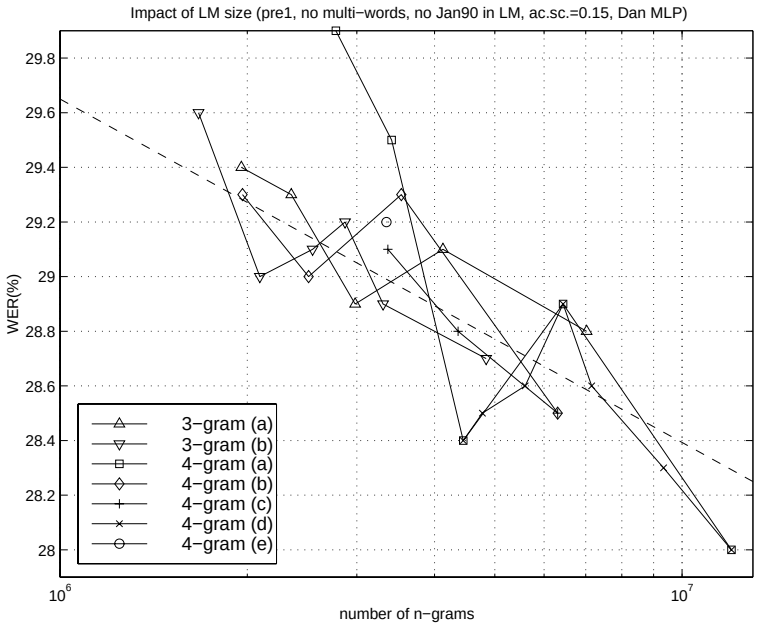


Figure 7.6: Summary of LM impact. (Dan MLP, "no Jan'90", 3/4-grams)

Before drawing any firm conclusions, it should be kept in mind, that the fluctuations due to different acoustic scaling factors can be up to 1%, and the difference between two LMs might vary up to 1%.

To allow a fair comparison between all LMs, the acoustic scaling factor should be optimized in each case, or the results should be averaged over a number of different values of acoustic scaling.

However, this could easily multiply the computations by a factor of 5-10. A way to work around

this problem could be to use a more severe pruning so that the decoding runs faster. However, before doing so, some of the experiments reported here (for example figure 7.4) should be repeated with this new pruning to see if the same conclusions can be drawn, i.e. if the WER as a function of acoustic scaling remains similar.

Now, returning to figure 7.6, the cut-offs for the different LMs are (from left to right on the figure):

3-gram (a): 5_5, 4_4, 3_3, 2_2, and 1_1

3-gram (b): 4_8, 3_6, 3_4, 2_4, 2_3, and 1_2

4-gram (a): 5_5_5, 4_4_4, 3_3_3, 2_2_2, and 1_1_1

4-gram (b): 4_8_12, 3_6_9, 2_4_6 and 1_2_3

4-gram (c): 3_4_5, 2_3_4 and 1_2_3

4-gram (d): 3_3_3, 2_3_3, 2_3_3, 2_2_2, 1_2_2, 1_1_2, and 1_1_1

4-gram (e): 2_4_8

From the graph for 3-gram (a) and 4-gram (a) it is seen that having all cut-offs=3 gives a good trade-off between number of parameters and WER. On the other hand, having a 4-gram with cut-offs=5_5_5 or 4_4_4 performs worse than other 3-/4-grams with a similar number of parameters.

For the rest of the points they all lie within a range of approximately ± 0.25% from a line drawn through all the data. This fluctuation is likely to depend on the test data and/or the chosen acoustic scaling factor and other recognition parameters.

7.7 MLP retraining, "Some Jan'90"

7.7.1 Results for MLP retrain 4

MLP retrain 4				
iteration	epoch	learn rate	# err	WER
boot	8 (final)	0.00025	1009	29.2 %
iter1	8 (final)	0.00025	963	27.9 %
iter2	6	0.001	1015	29.4 %
iter2	7	0.0005	993	28.7 %
iter2	8 (final)	0.00025	960	27.8 %
iter3	5	0.001	1032	29.9 %
iter3	6	0.0005	1006	29.1 %
iter3	7 (final)	0.00025	1004	29.1 %

Table 7.11: Results with MLPs from retrain 4: different iterations and different epochs. The LM was a "Some Jan'90" trigram with cut-offs=2. Default decoding parameters were used.

Table 7.11 shows the recognition results with MLPs from retrain 4. For each of the 4 iterations results are shown for the final MLP. The best results are obtained after iter1 (27.9%) and iter2 (27.8%). When proceeding with iter3, performance degrades to 29.1%, which can be due to over-training of the MLP.

The best result of 27.8% with retrain 4 compares to the 28.8% reported with the "Dan MLP" in table 7.7, which is a gain of 1% simply by retraining the MLP. Both MLPs have 1000 hidden units.

For iter2 and iter3, performance was also tested for the MLPs from the two epochs before the final epoch. In iter2 the WER drops 0.7% and 0.9% in the last two epochs. It could be interesting to see if the WER would drop further if more epochs were added with smaller learning rates. In iter3, WER is the same for the last two epochs, and further epochs is not likely to improve performance.

7.7.2 Results for MLP retrain 5

In this section, results are reported for MLPs from retrain 5. Recall that retain 5 only differs from retrain 4 by having 2000 hidden units in stead of 1000.

MLP retrain 5				
iteration	epoch	learn rate	# err	WER
boot	9 (final)	0.00025	930	26.9 %
it1	6	0.001	980	28.4 %
it1	7	0.0005	941	27.2 %
same, but flt.point MLP			941	27.2 %
it1	8 (final)	0.00025	947	27.4 %
same, but flt.point MLP			948	27.4 %
it2	5	0.001	988	28.6 %
it2	6	0.0005	969	28.0 %
it2	7 (final)	0.00025	941	27.2 %

Table 7.12: Results with MLPs from retrain 5: different iterations and different epochs. The LM was a "Some Jan'90" trigram with cut-offs=2. Default decoding parameters were used, except the two test that used floating point MLP programs.

The lowest WER with retrain 5 MLPs was 26.9% which was obtained already after the boot iteration. The two next iterations only get down to 27.2%, and in iter1, the WER even increases a bit to 27.4% after the last epoch.

Even though it is surprising that the best performance is after only one iteration, it is assuring that in all three iterations the WER gets down to about the same value. It is hard to say whether continuing with iter1 and iter2 cause an over-trained MLP, or whether the differences are simply due to statistical variance. It is also possible (likely) that different acoustic scaling should be used with the different MLPs.

Retrain 5, acoustic scaling		
acoustic scale	# err	WER
0.15	941	27.2 %
0.16	934	27.0 %
0.17	938	27.1 %
0.18	960	27.8 %
0.19	953	27.6 %
0.20	941	27.2 %
0.21	935	27.1 %
0.22	940	27.2 %
0.23	948	27.4 %
0.24	967	28.0 %
0.27	984	28.5 %

Table 7.13: Influence of acoustic scaling on WER with the final MLP from iter2 of retrain 5. The LM was a "Some Jan'90" trigram with cut-offs=2. Except for acoustic scaling, the decoding used default parameters.

Table 7.13 shows the WER for different values of acoustic scaling. These numbers are also found in figure 7.7.

Figure 7.7 shows WER as a function of acoustic scaling for the default "Dan MLP", and for the MLP after iter2 of retrain 5. The curve for the "Dan MLP" is quite smooth (at least with the

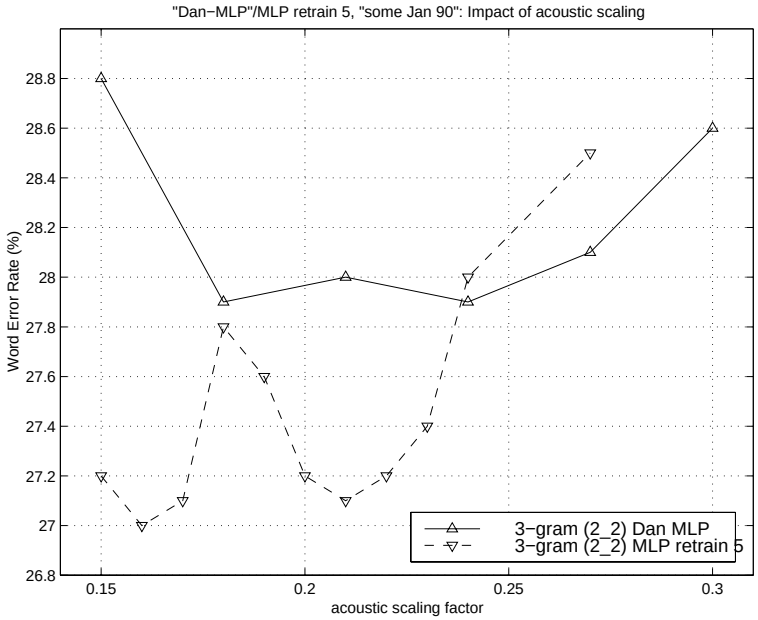


Figure 7.7: "no_bref" 3gr2_2 LM: Impact of acoustic scaling factor. (Dan MLP and MLP retrain 5, iter2)

data points at hand) and describes a wide U-curve. For retrain 5 the performance curve is somewhat strange, with a literal W-shape. There seem to be no plausible explanation for this large fluctuation.

The method of only varying the acoustic scaling is (can be) unfair to the larger values of acoustic scaling. When the beam remains fixed, lowering the acoustic scaling implies less pruning. The same would be true for the LM scaling, if it had been used (it is 1 in all the experiments).

There are two minima in figure 7.7: at 0.16 (global) and at 0.21 (local). However, if is the difference due to implicit less pruning at 0.16? To test this, experiments were repeated for acoustic scale = 0.21, but now with less pruning, by increasing the beam width.

Retrain 5			
beam	state beam	# err	WER
4 (def.)	5 (def.)	935	27.1 %
5	6	936	27.1 %
6	7	940	27.2 %

Table 7.14: Comparison of different pruning parameters. The LM was a "Some Jan'90" trigram with cut-offs=2. Acoustic scaling = 0.21 and varying beam values. The remaining decoding parameters were set to default values.

As table 7.14 shows, that increasing the search space with higher beam values does not reduce the WER.

At this point a "best combination" was tested, using the final MLP from the boot iteration of retrain 5, a 4-gram with cut-offs=1 ("Some Jan'90"), n_hyps increased to 14, and keeping the remaining parameters to default values. This setup gave 24.6% WER (851 errors) and was the best result obtained within the experiments using "Some Jan'90".

This compares to the "Dan MLP" that got 26.9% WER with the same LM, but using the default

n_hyps=7. (Going from n_hyps=7 to 14 showed about 1% improvement in an earlier experiment using an "All Jan'90" trigram.)

7.7.3 Results for MLP retrain 6

This section contains results from retrain 6 where the boot iteration uses a target file from FPMs for the MLP training instead of doing a forced alignment.

MLP retrain 6				
iteration	epoch	learn rate	# err	WER
boot	10 (final)	0.00025	983	28.4 %
it1	8 (final)	0.00025	965	27.9 %
it2	8 (final)	0.00025	925	26.8 %
it3	8 (final)	0.00025	950	27.5 %

Table 7.15: Recognition results with MLPs from retrain 6 using a default trigram trained on "Some Jan'90" and using default pruning/decoding.

The results from retrain 6 in table 7.15 shows a minimum WER (26.8%) after iter2 (at least when using the default decoding setup). This minimum compares to the 27.8% from retrain 4 (table 7.11). The two MLPs have the same size (1000 HU), but retrain 6 uses a different training scheme, because the boot iteration use a forced alignment (target file) from FPMs.

An explanation of the better performance can be that the MLP somehow learns a bit from both the first segmentation from FPMs and from the later re-segmentations. This might result in an MLP that generalizes better, and thereby improves WER.

7.7.4 Results for MLP retrain 6b

Retrain 6b is basically retrain 6, but with a fixed learning rate schedule, that decreases more slowly. Table 7.16 shows recognition results from retrain 6b.

MLP retrain 6b						
			"Some Jan'90"		"No Jan'90"	
iteration	epoch	learn rate	# err	WER	# err	WER
boot	14 (final)	0.00025	963	27.9 %	976	28.2 %
it1	13 (final)	0.00025	930	26.9 %	942	27.3 %
it2	12 (final)	0.00025	934	27.0 %	937	27.1 %
it3	13 (final)	0.00025	932	27.0 %	943	27.3 %

Table 7.16: Results from retrain 6b using a trigram trained on "Some Jan'90" and "No Jan'90", and default decoding parameters.

Retrain 6b was performed much later than retrain 6, after that "Some Jan'90" had been replaced by "No Jan'90". However, to allow comparison to retrain 6, recognition test were made for both LM training set. In both cases, the LM is a trigram with cut-offs=2.

When comparing to retrain 6 three conclusions can be made: a fixed slowly decreasing learning rate schedule (6b) does not perform better, but best results are obtained after only 2 iterations in stead of 3, and in retrain 6b the WER stays near the minimum, even if more training iterations are used.

The second comparison is between the LM training set. Here it is seen, that the "non-cheating" LM using "No Jan'90" has 0.3-0.4% higher WER than "Some Jan'90" – a result that could be expected.

7.7.5 Results for MLP retrain 7

Retrain 7 resembles retrain 4 in that the target labels for the boot iteration were created from a forced alignment. The difference being that retrain 7 uses the "Dan MLP" instead of the "Final MLP" obtained from FPMs.

MLP retrain 7						
			"Some Jan'90"		"No Jan'90"	
iteration	epoch	learn rate	# err	WER	# err	WER
boot	8 (final)	0.00025	962	27.8 %	966	28.0 %
it1	8 (final)	0.00025	979	28.3 %	985	28.5 %
it2	8 (final)	0.00025	943	27.3 %	958	27.7 %

Table 7.17: Results from retrain 7 using a trigram trained on "Some Jan'90" and "No Jan'90", and default decoding parameters.

If table 7.17 is compared to retrain 4, one sees that using a better MLP for the first forced alignment ("Dan MLP" in retrain 7), decreases WER with 0.5% (from 28.7% to 27.3% both using "Some Jan'90").

The WER after each iteration in retrain 7 describes a little strange behavior as it increases in iter1, and then obtain it's minimum in iter2. This is true for both LM training sets. It would be interesting to see what effect one (or several) more iteration might have.

As for retrain 6b, the "Some Jan'90" LM performs better than the "No Jan'90" LM. There is a difference of 0.2-0.4%.

7.8 MLP retraining, "No Jan'90"

From MLP retrain 8 and on, LMs trained on "No Jan'90" were used. However, for comparison some of the previous sections have also shown results with "No Jan'90" LMs.

7.8.1 Results for MLP retrain 8

Retrain 8 uses the same training scheme as retrain 6, but now with 2000 HU in stead of 1000 HU.

MLP retrain 8				
iteration	epoch	learn rate	# err	WER
boot	9 (final)	0.00025	966	28.0 %
it1	9 (final)	0.00025	931	26.9 %
it2	8 (final)	0.00025	896	25.9 %
it3	8 (final)	0.00025	914	26.4 %

Table 7.18: Recognition results with MLPs from retrain 8 using a default trigram trained on "No Jan'90" and using default pruning/decoding.

Since retrain 6 have no results with "No Jan'90" LMs, retrain 6b (with fixed learning rate schedule) is used for comparison. The best performance goes down from 27.1% WER (table 7.16) to 25.9%, which is a gain of 1.2% for doubling the number of MLP parameters from 1000 HU to 2000 HU.

The immediate question is: "what if an even bigger MLP is used?", which the next section will answer (retrain 9).

If table 7.18 is compared to retrain 5 (also 2000 HU) retrain 8 wins again (even using a weaker "No Jan'90" LM). Using the FPMs segmentation in the boot iteration decreases WER from 27.2% to 25.9% (1.3% abs).

It thus seems that these two things improve performance independently (and in an additive manner): 1) increasing the MLP size, and 2) using another segmentation file for the boot iteration.

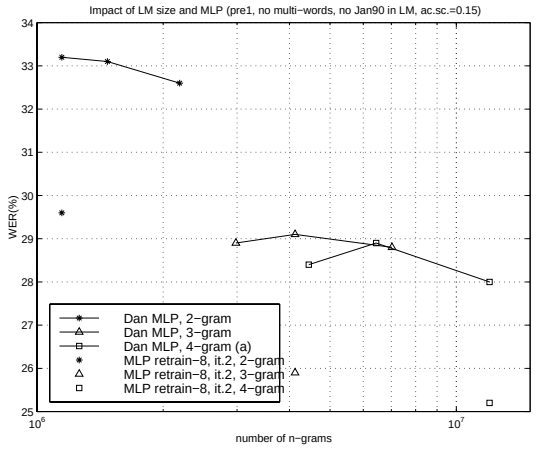


Figure 7.8: WER as a function of number of LM parameters for the "Dan MLP" and retrain 8, iter2, using LMs trained on "No Jan'90". Decoding was done with the default parameters.

Figure 7.8 shows a comparison between the default "Dan MLP" and the MLP after iter2 of retrain 8 using a bigram, trigram or 4-gram trained on "No Jan'90". For the "Dan MLP" results are shown for LMs where all cut-offs were 3, 2, or 1 respectively from left to right. For retrain 8 recognition was only performed with one bi-, tri-, and 4-gram.

The new MLP is seen to perform 2.8-3.6% better than the "Dan MLP" depending on the LM (when using the default decoding parameters), and the new MLP gain most with the bigram, and least with the 4-gram. However, the relative improvement remains almost the same (10-11% rel.).

Figure 7.8 compared two MLPs using the acoustic scaling (default=0.15), which might not be fair. Figure 7.9 shows how WER changes as a function of the acoustic scaling for the two MLPs, using 3 different LMs.

The two lower solid lines ("Dan MLP, 3-/4-grams) in figure 7.9 might be recognized as they were also shown in a previous comparison (figure 7.4). The bi-, tri-, and 4-grams used cut-offs of 3, 2, and 1 respectively, and were all trained on "No Jan'90".

The first over-all observation is that the shape of the solid lines are alike, and so are the shape of dashed lines. For both MLPs the bigram is lacking far behind the 3- and 4-gram, and the 4-gram wins slightly over the trigram.

There is an important difference between the dashed and solid curves. The dashed curve (retrain 8, iter2) have a clear minimum around 0.15. The solid curves on the other hand does not have one clear minimum, but have a long flat area with minimums occurring between 0.17 and 0.27.

The 6 points that were compared in figure 7.8 correspond to the WERs at acoustic scale=0.15 in figure 7.9, and it is seen that the "Dan MLP" (solid line) is disfavored.

When taken using the WER for the optimal acoustic scaling, the new MLP gain 2.4% (32.0%-29.6%) with the bigram, 2.2% (28.1%-25.9%) with the trigram, and 2.0% (27.1%-25.1%) with the 4-gram. Again the relative improvement due to the new MLP remains almost constant (7.4-7.8% rel.)

For the "Dan MLP" there are some fluctuations in the curves, but for retrain 8, iter2 it seems that the acoustic scaling factor can be optimized for one LM, and will then also be (near) optimal for

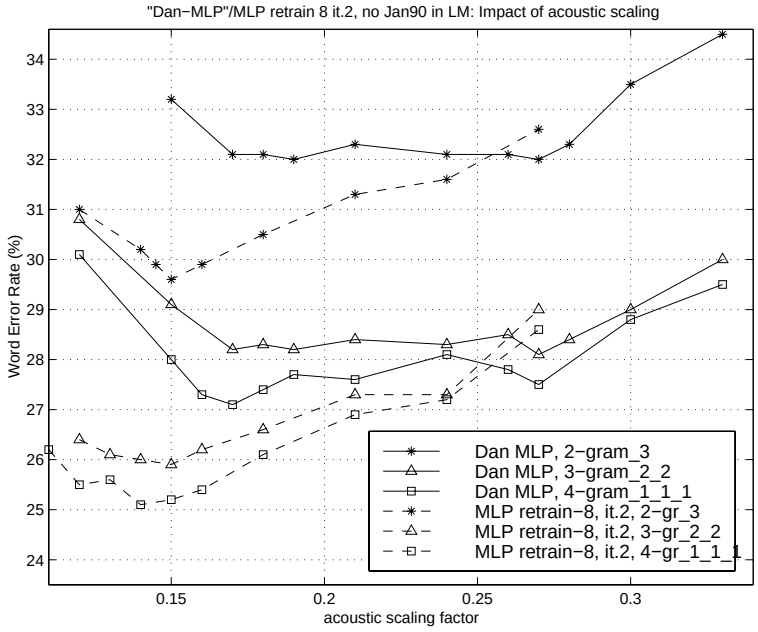


Figure 7.9: WER as a function of acoustic scaling for two MLPs: retrain 8, iter2 and the "Dan MLP", and for three different LMs trained on "No Jan'90".

other LMs.

What is the explanation of the difference in shape between the solid and dashed curves? It was earlier shown for the "Dan MLP" that for acoustic scale=0.21, WER didn't improve if the the beam widths were increased, meaning that the beams were not a limiting factor for the "Dan MLP". However, could it be that for retrain 8, iter2, the beam size is a limiting factor causing increasing WER for acoustic scalings above 0.15? To test this hypothesis a recognition was performed with the MLP from retrain 8, iter2, using acoustic scaling=0.24 and increasing the beam and state beam from their default values of 4 and 5 respectively.

MLP retrain 8, iter2, ac.sc.=0.24, "No Jan'90"				
LM	beam	state beam	# err	WER
4-gram, c.o.=1_1_1	4 (def.)	5 (def.)	940	27.2 %
4-gram, c.o.=1_1_1	5	6	938	27.1 %
4-gram, c.o.=1_1_1	6	7	938	27.1 %
3-gram, c.o.=2_2	4 (def.)	5 (def.)	944	27.3 %
3-gram, c.o.=2_2	5	6	942	27.3 %

Table 7.19: Recognition with various beam sizes. LMs were trained on "No Jan'90", and the decoding used acoustic scaling of 0.24; unmentioned parameters used default values.

Table 7.19 shows that the default beam and state beam are not limiting factors in the case of retrain 8, iter2, ac.sc.=0.24.

Another theory to explain the shape of the dashed curves in figure 7.9 might be that the MLP in retrain 8, iter2 is over-trained. The MLP might start to "remember" the training data and have output probabilities that become closer to 0.0 or 1.0.

Take the example of an MLP with two output 'a' and 'b', and say that the "true" a posterior probabilities given the acoustic x are:

$$P(q=a|x)=0.2 \quad P(q=b|x)=0.8, \tag{7.3}$$

which is a proportion of 1:4, and suppose that the probability estimate output by the MLP is "over confident" in 'b' by having the values:

$$\hat{P}(q=a|x)=0.1 \quad \hat{P}(q=b|x)=0.9, \tag{7.4}$$

which is 1:9. Then to get back to the "true" 1:4 one need to raise the estimates to the power of 0.631:

$$\frac{\hat{P}(q=a|x)^p}{\hat{P}(q=b|x)^p} = 1/4 \Rightarrow \tag{7.5}$$

$$(\frac{\hat{P}(q=a|x)}{\hat{P}(q=b|x)})^p = 1/4 \Rightarrow \tag{7.6}$$

$$(1/9)^p = (1/4) \Rightarrow \tag{7.7}$$

$$p = ln(1/4)/ln(1/9) = 0.631 \tag{7.8}$$

Then one gets the relation:

$$P(q=a|x) = c\cdot\hat{P}(q=a|x)^{0.631} = c\cdot 0.234 \tag{7.9}$$

$$P(q=b|x) = c\cdot\hat{P}(q=b|x)^{0.631} = c\cdot 0.936 \tag{7.10}$$

and the proportion is back to the "true" 1:4. Remember that a constant scaling factor as c doesn't influence on the decoding, so simply raising the MLP probability estimates are sufficient.

In practise the value of p has to be found by recognition experiments, e.g. as summarized in figure 7.9.

An a analysis of the distribution of probability estimates from various MLPs could confirm or reject this theory. The entropy calculated over all frames in the test data could be one measure to predict if an MLP is over-trained. See the section on future work for more details.

Less pruning/ best system

At the end of this work, the MLP from iter2 of retrain 8 was found to be the best performing, and it was decided to try to improve WER further. In earlier experiments it was shown, that increasing n_hyps could improve recognition without an explosion in computations.

retrain8, iter2: effect of n_hyps			
n_hyps	speed (× real-time)	# err	WER
7 (def.)	17	872	25.2 %
14	23	844	24.4 %
28	43	804	23.3 %

Table 7.20: Performance of the best system: MLP from retrain 8, it.2, ac.sc.=0.15

Table 7.20 shows results, where n_hyps was first doubled to 14 (gaining 0.8%) and then to 28 (gaining 1.1%) resulting in 23.3% WER, which was the very best result obtained in all the experiments.

The cost of the above improvements is a moderate increase in computation time, and a slight increase in process size to a maximum of about 400 MB ram.

In most of the other experiments, a set of default decoding parameters was used. It is possible that these parameters can be optimized by increasing n_hyps, and tightening some of the other pruning

parameters, keeping the same speed, but reducing WER. This could be determined by a series of experiments with various prunings and focus on the trade-off between speed and WER.

Another study could be concerned with the required memory, which is as much a limiting factor as the computation time.

7.8.2 Results for MLP retrain 9

Retrain 9 repeats retrain 4 and 8, but now the size of the MLP has grown to 4000 HU.

MLP retrain 9				
iteration	epoch	learn rate	# err	WER
boot	10 (final)	0.00025	918	26.6 %
it1	9 (final)	0.00025	926	26.8 %
it2	7 (final)	0.00025	950	27.5 %

Table 7.21: Results with MLPs from retrain 9 using default decoding parameters and a trigram trained on "No Jan'90" (with cut-offs=2).

Table 7.21 contains recognition results with retrain 9 MLPs. The best performance is obtained after only one iteration, although that during training frame accuracies are much higher in the last two iterations.

This mismatch between frame accuracy and WER has been observed before. One could suspect that either is 4000 hidden units too big an MLP for the training data at hand, or that the MLP training is sub-optimal. It is possible that the fixed learning rate schedule would give better WER results.

It is also worth to mention that in retrain 8 three iterations were needed to reach the minimum WER. The best WER with retrain 9 is 26.6% which is 0.7% worse than retrain 8 for the default acoustic scale of 0.15. One should remember that this acoustic scaling was optimal for retain 8 (see figure 7.9) and thus might be an advantage to retrain 8, although the optimal acoustic scaling for retrain 9 can be expected to be close to 0.15 as well.

The conclusion can be that for the amount of acoustic training data used in these experiments, the optimal MLP size is 2000 HU (retrain 8) at least with the training technique used at present.

7.8.3 Results for MLP retrain 10

Retrain 10 uses a fixed learning rate schedule, where the learning rate decreases more slowly than the default. The segmentation for the boot iteration was that used in iter1 of retrain 6b, but in retrain 10 training started from scratch, whereas iter1 in retrain 6b continued from the MLP after one epoch of retrain 6b boot iteration.

If looking at the results with the default acoustic scaling of 0.15, the best result with retrain 10 is 27.0% after iter1, which compares to 27.1% which was obtained with retrain 6b after iter2, also on "No Jan'90". (Remember that the two iterations correspond to each other.) The 0.1% difference is not significant, and also there were not performed experiment with different acoustic scaling for retrain 6b.

For retrain it was decided to run experiments with a few different values of acoustic scaling. All the curves in figure 7.10 have the same shape with minimum WER for acoustic scale=0.18 except for the final MLP in iter2. But in all cases near optimal WER is obtained with acoustic scalings close to 0.18. The optimum acoustic scaling of 0.18 is higher than for retrain 8, where 0.15 was the best value. A premature conclusion on this difference is that for larger MLPs a smaller acoustic scaling is needed.

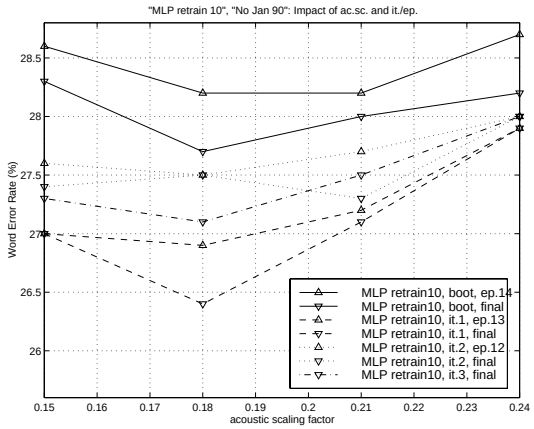


Figure 7.10: Results with final MLPs from each iteration in retrain 10 as well as the MLPs from 3rd last epoch in the first three iterations. WER is shown as a function of acoustic scaling. Default values were used for the remaining parameters, and the LM was a "no Jan'90" trigram with cut-offs=2.

7.8.4 Results for MLP retrain 11

Retrain 11 compares to retrain 7 in the way that retrain 11 used the same fixed learning rate schedule as retrain 10 which decreases the learning rate slower than the default schedule.

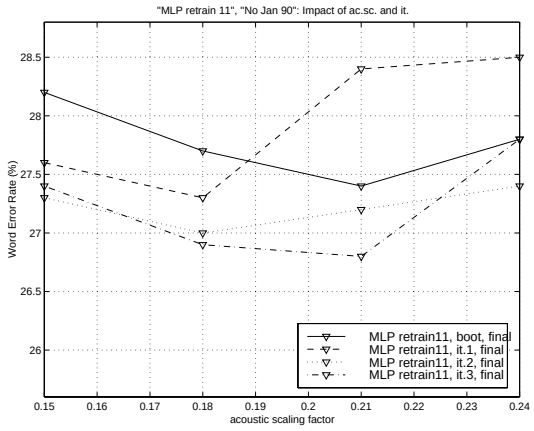


Figure 7.11: Results with final MLPs from each iteration in retrain 11 using various values of acoustic scaling. Default values were used for the remaining parameters, and the LM was a "no Jan'90" trigram with cut-offs=2.

In figure 7.11 the WER for the default acoustic scaling of 0.15 can be compared to the "No Jan'90" values obtained with retrain 7. It shows that the best result improves from 27.7% to 27.3% when the new learning rate schedule is used.

The curves in figure 7.11 do not have quite as nice shapes as in figure 7.10 (e.g. curves are crossing each other). But still the minimum WER is obtained for acoustic scaling in the range 0.18-0.21.

Chapter 8

Conclusion and Future work

8.1 Summary

8.1.1 LM text preprocessing

1.0% WER improvement was obtained by replacing the LM text preprocessing with the new "pre1": the WER goes down from 28.3% to 27.3%, when in both cases using a trigram with cut-offs=3, and trained on "All Jan'90".

When the LM training set is changed from "All Jan'90" to "No Jan'90" (no cheating) WER goes up from 27.3% to 28.9% when using a trigram with cut-offs=3. It is also observed that when using "No Jan'90", large LMs provide less improvements than was the case when "cheating" ("All Jan'90").

With pre1 preprocessing, the addition of 500 multi-words consistently improved WER with about 0.5%, when WER was plotted against the number of parameters in the LM. This was without retraining the pronunciations for the multi-words, and is thus only the LM effect of adding multi-words.

8.1.2 Different LM size

A large collection of LMs were tested. The most remarkable improvement is when going from a bigram to a trigram: approximately going from 33% to 29% WER ("No Jan'90").

For the trigrams and 4-grams the WER as a function of number of parameters shows a slowly deceasing function with some deviation. With the number of parameters on a logarithmic scale, the WER approximately decreases linearly with 1.25% per decade with deviations of about $\pm 0.25\%$.

Leaving out the most deviating point, the WER varies between 28.3% and 29.5% for LMs with 2M-10M parameters.

8.1.3 Acoustic scaling

It was shown, that the acoustic scaling plays an important role, and that WER can easily vary 1-2% if the acoustic scaling is varied within 0.15-0.30 which are sensible values.

The WER as a function of acoustic scaling was found to mostly depend on the MLP. But even for a fixed MLP, the difference in WER between two LMs can vary up to 1% with acoustic scaling between 0.15 and 0.30.

8.1.4 MLP retraining

A simple MLP retraining (retrain 4) improves WER from 28.8% to 27.8% when using a trigram with cut-offs=2, trained on "Some Jan'90".

When increasing the MLP size from 1000 hidden units (retrain 4) to 2000 hidden units (retrain 5) the WER goes further down from 27.8% to 26.9%.

In retrain 6 a new training scheme was introduced, using one segmentation for the boot iteration, and continuing with the usual forced alignment in following iterations. This way the 27.8% from retrain 4 is improved to 26.8%, still using a trigram with cut-offs=2, trained on "Some Jan'90".

Repeating retrain 6 with a new fixed learning rate schedule (retrain 6b) didn't improve WER: 26.9% (with the same LM, but trained on "No Jan'90": 27.1%).

In retrain 4 (and 5) the first forced alignment was made with a supposedly not well-trained MLP. When replacing this MLP with the "Dan MLP" (retrain 7), WER goes down from 27.8% to 27.3% using the "Some Jan'90" trigram (27.7% with the "No Jan'90" trigram) with cut-offs=2.

Repeating retrain 6 with a bigger MLP (retrain 8, 2000 hidden units) the WER goes further down to 25.9% (now using "No Jan'90"). Increasing the size further to 4000 hidden units (retrain 9), performance degrades to 26.6% WER, which might indicate over-training of the MLP; 4000 hidden units might be too big an MLP given the size of the training data.

Retrain 11 use the new learning rate schedule, but is otherwise as retrain 7, and decreases WER from 27.7% to 27.3% using the "No Jan'90" trigram.

8.1.5 Decoding pruning

0.8% WER improvement was obtained by increasing n_hyps from 7 to 14, at the cost of 1.35-1.9 longer computation time. This improvement was obtained in two quite different experiments: 1) using the "Dan MLP" with a trigram with cut-offs=2, trained on "All Jan'90", in which case the WER goes from 26.3% to 25.4%, and 2) using the (best) retrain 8, iter2 MLP with a 4-gram with cut-offs=1 (the best), trained on "No Jan'90", in which case the WER goes from 25.2% to 24.4%. In this last setup, another doubling of the value of n_hyps to 28, the WER goes further down to 23.3%, which was the best result obtained in all the test.

8.2 Conclusion

The overall conclusion is that

- careful preprocessing of LM text is necessary
- 500 multi-word improves performance (even without retraining the pronunciations)
- Leaving out the BREF months increases WER
- trigrams are far better than bigrams
- 4-grams are (can be) a little better than trigrams
- acoustics scaling is important
- Careful MLP retraining can gain 2% in WER
- a larger MLP (2000 HU) improves additionally 1%
- using less pruning can give 2% WER reduction.

The best result obtained was 23.3% WER.

All these improvement were obtained with the same data material as was used for the initial system. It is expected that more acoustic and text training data will help improve results further.

8.3 Future Work

There is nothing as easy as to propose work to be done (by others...), so in this section, a collection of research ideas is presented. All the suggestions for future work are aimed at improving the French recognition system.

8.3.1 Lexicon training on multi-words

In the multi-word experiments pronunciations for multi-words were obtained as the Cartesian product of the composing words. Lexicon training on the multi-words is likely to capture inter-word dependencies, and provide context dependent pronunciations. One should be careful with pronunciation pruning on multi-words, because a 4-tuple is likely to have more valid pronunciations than a single word. (The same might apply to long and short words; long words are more likely to have more valid pronunciations)

8.3.2 Pre2 preprocessing without bug

A bug was discovered in the pre2 preprocessing: the '-' in "t- il" was accidentally skipped together with all the other '-' in the text, making it impossible to distinguish 't-' from the letter 't' which has another pronunciation (/t/ and /te/ respectively.

It might be interesting to see how pre2 performs without this bug, and also to test pre2 with more multi-words, (e.g. 1000).

8.3.3 Optimize LM training parameters

The CMU/Cambridge LM Toolkit provides a range of options that have not been used in this work. It is possible that recognition results can be improved by tuning some of these parameters.

8.3.4 More or less LM training text

When starting on the work described in this report, one of the goals were to train LMs on more text data. However, this has not been done yet, but should be a manageable task, that should take little human effort, but a lot of disk-space and memory.

It will also be interesting to train LMs on only a few months of "Le Monde" close to the "BREF months". Maybe it is better to train LMs targeted to a specific task.

The raw text from "Le Monde" also contain key-words and other information about the text. This could be used to train specific LMs for specific topics, e.g. an LM for international politics.

8.3.5 More speech data to train MLPs

More acoustic data to train the MLPs can be expected to improve recognition. To get an idea of how much improvement can be expected, one can first go in the other direction, and repeat one of the MLP training experiments, but only using half the training data. If this causes a significant raise in WER, then one might extrapolate, and expect that more training data could improve recognition.

8.3.6 PLP features to improve BREF results

In experiments at FPMs it was shown that pure PLP features performed better on BREF than RASTA-PLP. This is not surprising since BREF is clean studio speech, and RASTA is designed to be robust towards different channels and noise. But for real-world applications like broad-cast speech recognition RASTA-PLP is probably still be the right choice.

8.3.7 Use delta delta features

This repost describes several experiments where the size (of the hidden layer) of the MLP is increased and causing lower WER. Another way to increase the MLP size is to have more input parameters. Delta-delta features are often used in large HMM/ANN systems, but have not yet been tested in the context of this work.

8.3.8 Divide MLP output by transition probabilities

In the theory section, it was argued that it might not be a good idea to divide the MLP output by state priors. In stead it was suggested that the local posteriors could be divided by (prior) *transition* probabilities. However, it should be studied how these transition probabilities would (or would not) cancel out with transition probabilities in the HMMs.

Another possibility is to divide by state priors that have been raised to a power p smaller than 1 (e.g. $p = 0.25$)).

8.3.9 Find best trade-off between pruning parameters

Some of the experiments described in this report, showed the effect of the pruning parameters. A more close study of the pruning parameters could be used to find the best trade-off e.g. for a real-time system, and/or a 10×real-time system.

8.3.10 Analysis of word errors

An analysis of the word errors might reveal if certain words or phonemes are causing many errors. This might give new ideas for for improvements. One possibility that was mentioned in this report was to try to have more "trouble words" in the multi-words, hoping that the extended context could improve recognition of those words.

8.3.11 Analysis of MLP output

In the MLP training sections and recognition sections it was suggested that some of the MLPs were over-trained. It was postulated that one sign of over-training was that the local probability estimates output by the MLP gets closer to zero and one. A study of the distribution of the MLP-output for different MLPs might confirm or reject this postulate.

Such a study could also compute sample error rates at the frame-level, for small intervals of values of the MLP-output. That is, calculate the frame accuracy for each interval. For example: count all number of times an MLP-output was within an interval (e.g. 0.10-0.11), and count the number of times that output was the "right" (determined by a forced alignment). If the MLP output "true" probabilities, then outputs within 0.10-0.11 should be the correct state 10.5% of the time. This means that ideally the above sample frame accuracy should be a linear function of the MLP-output. If it is not, but is an increasing function, the MLP-output could be put through the inverse function to get a straight line. This might be better than to use acoustic scaling. (and should be combined with some of the ideas about division by transition probabilities.)

The above study can be made with all phonemes in one pool, or better, separately for each phoneme. This would mean that a separate weighting function could be derived for each phoneme (possibly followed by a normalization to get the probabilities to sum to one).

Because it is the relative size of probabilities that matters in the decoding (not the absolute values), the above analysis should be done on a logarithmic scale, e.g. having the interval equally distributed on a logarithmic scale. If the corresponding sample error-rates are also shown on a logarithmic scale, "true" probabilities will still be represented by a line.

The inverse function(s) most likely should also have a logarithmic nature, being more accurate for small probabilities than for large probabilities.

8.3.12 Continued lexicon training

The same pronunciation dictionary has been used for all the experiments in this report. It is possible that repeating the lexicon training with one of the new and better MLPs would make an improvement.

Appendix A

MLP retrain 4

Host: montfort.
Training period: August 16 - 26, 1998.
Train speed: 42-44.5 MCUPS (million connection updates per second).
CV speed: 143-213 MCPS (million connections per second).
3 hours and 50 minutes per epoch.

MLP retrain 4, frame accuracy												
	boot			iter1			iter2			iter3		
learn		fr. corr. (%)			fr. corr. (%)			fr. corr. (%)			fr. corr. (%)	
rate	ep	train	CV	ep	train	CV	ep	train	CV	ep	train	CV
-	-	-	3.12	-	-	67.36	-	-	69.77			
0.008	1	75.47	67.18	1	80.44	69.85	1	81.71	70.67	-	-	70.59
0.008	2	80.01	69.36	2	81.68	70.76	2	82.38	71.36	1	82.34	71.13
0.008	3	80.95	69.85	3	82.19	71.04	3	82.73	71.62	2	82.76	71.62
0.004	4	81.09	73.31	4	82.29	74.20	4	82.78	74.77	3	82.91	74.80
0.002	5	81.23	75.72	5	82.46	76.59	5	82.96	76.84	4	83.10	76.75
0.001	6	81.34	76.94	6	82.55	77.83	6	83.04	78.10	5	83.25	78.16
0.0005	7	81.39	77.63	7	82.63	78.53	7	83.12	78.75	6	83.35	78.86
0.00025	8	81.44	78.00	8	82.69	78.88	8	83.20	79.06	7	83.42	79.15

Table A.1: Frame accuracy for MLP retrain 4

Appendix B

MLP retrain 5

Host: nendaz.
Training period: August 17 - 31, 1998.
Train speed: 45 - 46 MCUPS.
CV speed: 225 - 235 MCPS.
7 hours and 25 minutes per epoch.

MLP retrain 5, frame accuracy									
	boot			iter1			iter2		
learn rate	ep	fr. corr. (%) train	CV	ep	fr. corr. (%) train	CV	ep	fr. corr. (%) train	CV
-	-	-	1.71						
0.008	1	76.43	67.77	-	-	67.81			
0.008	2	81.32	69.86	1	81.55	70.0	-	-	70.04
0.008	3	82.55	70.39	2	83.04	71.42	1	83.16	71.20
0.008	4	83.27	70.86	3	83.87	71.13	2	84.05	71.59
0.004	5	83.54	74.30	4	83.79	74.16	3	84.65	74.67
0.002	6	83.73	76.56	5	84.23	76.86	4	85.02	77.00
0.001	7	83.84	77.90	6	84.44	78.54	5	85.21	78.76
0.0005	8	83.94	78.69	7	84.59	79.38	6	85.33	79.69
0.00025	9	84.00	79.04	8	84.68	79.79	7	85.44	79.95

Table B.1: Frame accuracy for MLP retrain 5

Appendix C

MLP retrain 6

Host: montfort.
Training period: August 29 - September 3, 1998.
Train speed: 41 - 45 MCUPS.
CV speed: 176 - 212 MCPS.
3:50 hours per epoch.

MLP retrain 6, frame accuracy												
	boot			iter1			iter2			iter3		
learn		fr. corr. (%)			fr. corr. (%)			fr. corr. (%)			fr. corr. (%)	
rate	ep	train	CV	ep	train	CV	ep	train	CV	ep	train	CV
-	-	-	3.04									
0.008	1	74.12	65.66									
0.008	2	78.52	67.30	-	-	67.07	-	-	69.46	-	-	70.26
0.008	3	79.40	67.92	1	80.15	69.43	1	81.63	70.32	1	82.37	71.28
0.008	4	79.88	68.47	2	81.40	70.65	2	82.36	71.79	2	82.78	72.19
0.008	5	80.18	68.65	3	81.95	71.07	3	82.78	71.87	3	83.06	72.17
0.004	6	79.94	72.13	4	82.07	74.02	4	82.84	74.74	4	83.00	74.80
0.002	7	80.01	73.96	5	82.24	76.52	5	83.00	76.96	5	83.21	77.01
0.001	8	80.02	75.11	6	82.33	77.71	6	83.08	78.18	6	83.34	78.28
0.0005	9	80.04	75.74	7	82.41	78.40	7	83.18	78.84	7	83.46	78.94
0.00025	10	80.06	76.00	8	82.46	78.73	8	83.26	79.15	8	83.53	79.27

Table C.1: Frame accuracy for MLP retrain 6

Appendix D

MLP retrain 6b

Host: nendaz.
Training period: September 28 - October 9, 1998.
Train speed: 44 MCUPS.
CV speed: 196 - 213 MCPS.
3:50 hours per epoch.

MLP retrain 6b, frame accuracy												
	boot			iter1			iter2			iter3		
learn rate	ep	fr. corr. (%) train	CV	ep	fr. corr. (%) train	CV	ep	fr. corr. (%) train	CV	ep	fr. corr. (%) train	CV
-	-	-	3.04									
0.008	1	74.12	65.66	-	-	67.00						
0.008	2	78.52	67.30	1	80.13	69.27	-	-	69.31			
0.008	3	79.40	67.92	2	81.35	70.65	1	81.53	70.59	-	-	70.52
0.008	4	79.88	68.47	3	81.91	71.02	2	82.27	71.80	1	82.26	71.47
0.0056	5	80.06	70.21	4	82.24	72.72	3	82.78	73.53	2	82.86	73.80
0.0040	6	80.28	71.94	5	82.53	74.36	4	83.14	74.76	3	83.33	75.07
0.0028	7	80.37	73.07	6	82.70	75.62	5	83.35	76.22	4	83.56	76.12
0.0020	8	80.44	73.95	7	82.80	76.55	6	83.51	77.09	5	83.74	77.17
0.0014	9	80.48	74.63	8	82.85	77.28	7	83.58	77.82	6	83.82	77.88
0.0010	10	80.52	75.16	9	82.91	77.83	8	83.64	78.32	7	83.90	78.41
0.0007	11	80.52	75.51	10	82.97	78.21	9	83.70	78.71	8	83.93	78.82
0.0005	12	80.54	75.78	11	83.01	78.48	10	83.74	78.97	9	84.01	79.10
0.00035	13	80.53	75.99	12	83.04	78.67	11	83.77	79.17	10	84.04	79.30
0.00025	14	80.55	76.11	13	83.08	78.80	12	83.82	79.29	11	84.09	79.43
0.000175										12	84.12	79.51
0.000125										13	84.15	79.59

Table D.1: Frame accuracy for MLP retrain 6b

Appendix E

MLP retrain 7

Host: nendaz.
Training period: September 2 - 6, 1998.
Train speed: 44 MCUPS.
CV speed: 178 - 213 MCPS.
3:50 hours per epoch.

MLP retrain 7, frame accuracy									
	boot			iter1			iter2		
learn rate	ep	fr. corr. (%) train	CV	ep	fr. corr. (%) train	CV	ep	fr. corr. (%) train	CV
-	-	-	3.11	-	-	67.43	-	-	69.09
0.008	1	75.50	67.34	1	80.52	69.21	1	81.71	70.20
0.008	2	80.26	69.65	2	81.70	70.78	2	82.40	71.38
0.008	3	81.28	69.97	3	82.29	71.08	3	82.73	71.51
0.004	4	81.52	73.35	4	82.39	74.29	4	82.80	74.55
0.002	5	81.71	75.83	5	82.57	76.44	5	82.98	76.73
0.001	6	81.81	77.25	6	82.69	77.77	6	83.07	77.97
0.0005	7	81.85	77.99	7	82.76	78.48	7	83.16	78.65
0.00025	8	81.92	78.37	8	82.83	78.82	8	83.24	78.98

Table E.1: Frame accuracy for MLP retrain 7

Appendix F

MLP retrain 8

Host: nendaz.
Training period: September 7 - 18, 1998.
train speed: 45 - 46 MCUPS.
CV speed: 193 - 232 MCPS.
7:30 hours per epoch.

MLP retrain 8, frame accuracy												
	boot			iter1			iter2			iter3		
learn rate	ep	fr. corr. (%) train	CV	ep	fr. corr. (%) train	CV	ep	fr. corr. (%) train	CV	ep	fr. corr. (%) train	CV
-	-	-	1.69	-	-	67.71						
0.008	1	75.06	66.36	1	81.46	69.88	-	-	69.88	-	-	70.89
0.008	2	79.89	68.02	2	82.97	70.57	1	83.07	70.93	1	83.98	71.49
0.008	3	81.14	68.81	3	83.75	71.11	2	84.00	71.51	2	84.64	72.21
0.008	4	81.83	69.27	4	84.26	71.58	3	84.59	71.87	3	85.08	72.25
0.004	5	81.99	72.39	5	84.52	75.02	4	85.02	75.29	4	85.47	75.48
0.002	6	82.20	74.40	6	84.78	77.26	5	85.35	77.70	5	85.80	77.79
0.001	7	82.29	75.79	7	84.91	78.60	6	85.53	78.98	6	85.99	79.04
0.0005	8	82.35	76.55	8	85.03	79.30	7	85.67	79.64	7	86.14	79.73
0.00025	9	82.35	76.94	9	85.13	79.64	8	85.77	80.00	8	86.25	80.03

Table F.1: Frame accuracy for MLP retrain 8

Appendix G

MLP retrain 9

Host: montfort.
Training period: September 11 - 30, 1998.
Train speed: 46 MCUPS.
CV speed: 155 - 244 MCPS.
14:30 - 14:40 hours per epoch.

MLP retrain 9, frame accuracy									
	boot			iter1			iter2		
learn rate	ep	fr. corr. (%) train	CV	ep	fr. corr. (%) train	CV	ep	fr. corr. (%) train	CV
-	-	-	2.23						
0.008	1	75.60	66.86	-	-	68.15			
0.008	2	80.59	68.43	1	81.97	70.87			
0.008	3	82.03	69.48	2	83.64	71.39	-	-	70.89
0.008	4	82.92	69.99	3	84.62	72.36	1	83.74	72.10
0.008	5	83.56	70.15	4	85.29	72.49	2	84.80	71.82
0.004	6	84.00	72.95	5	85.98	75.67	3	85.02	75.35
0.002	7	84.32	74.89	6	86.51	77.78	4	85.79	77.55
0.001	8	84.46	76.22	7	86.73	79.01	5	86.19	79.23
0.0005	9	84.54	77.05	8	86.88	79.79	6	86.42	80.11
0.00025	10	84.57	77.50	9	87.01	80.17	7	86.55	80.55

Table G.1: Frame accuracy for MLP retrain 9

Appendix H

MLP retrain 10

Host: montfort.
Training period: October 12 - 22, 1998.
Train speed: 43 - 44.5 MCUPS.
CV speed: 165 - 214 MCPS.
3:50 hours per epoch.

MLP retrain 10, frame accuracy												
	boot			iter1			iter2			iter3		
learn		fr. corr. (%)			fr. corr. (%)			fr. corr. (%)			fr. corr. (%)	
rate	ep	train	CV	ep	train	CV	ep	train	CV	ep	train	CV
-	-	-	3.05									
0.008	1	75.39	67.36	-	-	67.37						
0.008	2	80.05	69.23	1	80.29	69.41	-	-	69.35			
0.008	3	81.03	69.91	2	81.50	70.44	1	81.54	70.42	-	-	70.32
0.008	4	81.55	70.22	3	82.15	71.07	2	82.31	71.07	1	82.27	71.27
0.0056	5	81.80	72.65	4	82.57	72.71	3	82.82	73.19	2	82.87	73.68
0.0040	6	82.06	74.20	5	82.92	74.37	4	83.22	74.76	3	83.35	74.84
0.0028	7	82.19	75.30	6	83.08	75.64	5	83.46	76.03	4	83.61	76.00
0.0020	8	82.30	76.15	7	83.19	76.58	6	83.60	76.97	5	83.77	76.97
0.0014	9	82.37	76.90	8	83.26	77.39	7	83.69	77.67	6	83.86	77.76
0.0010	10	82.42	77.41	9	83.33	77.97	8	83.75	78.23	7	83.94	78.26
0.0007	11	82.47	77.78	10	83.36	78.40	9	83.79	78.65	8	83.99	78.68
0.0005	12	82.50	78.06	11	83.40	78.69	10	83.85	78.92	9	84.06	79.02
0.00035	13	82.52	78.27	12	83.43	78.90	11	83.88	79.14	10	84.09	79.24
0.00025	14	82.54	78.40	13	83.47	79.05	12	83.92	79.29	11	84.13	79.39
0.000175	15	82.57	78.50	14	83.50	79.16	13	83.95	79.42	12	84.16	79.49
0.000125	16	82.59	78.56	15	83.52	79.24	14	83.98	79.51	13	84.20	79.56

Table H.1: Frame accuracy for MLP retrain 10

Appendix I

MLP retrain 11

Host: nendaz.
Training period: October 12 - 22, 1998.
Train speed: 44.5 MCUPS.
CV speed: (34 -) 191 - 211 MCPS.
3:50 hours per epoch.

MLP retrain 11, frame accuracy												
	boot			iter1			iter2			iter3		
learning		fr. corr. (%)			fr. corr. (%)			fr. corr. (%)			fr. corr. (%)	
rate	ep	train	CV	ep	train	CV	ep	train	CV	ep	train	CV
-	-	-	3.11									
0.008	1	75.50	67.34	-	-	67.22						
0.008	2	80.26	69.65	1	80.30	69.05	-	-	68.96			
0.008	3	81.28	69.97	2	81.53	70.68	1	81.48	70.06	-	-	70.00
0.008	4	81.81	70.31	3	82.15	70.87	2	82.23	71.37	1	82.19	70.89
0.0056	5	82.07	72.35	4	82.55	72.76	3	82.74	73.15	2	82.79	73.39
0.0040	6	82.29	74.16	5	82.84	74.35	4	83.15	74.53	3	83.24	74.66
0.0028	7	82.38	75.29	6	83.03	75.81	5	83.35	75.86	4	83.50	75.86
0.0020	8	82.47	76.27	7	83.14	76.63	6	83.51	76.84	5	83.65	76.88
0.0014	9	82.55	77.01	8	83.21	77.40	7	83.57	77.60	6	83.75	77.71
0.0010	10	82.59	77.49	9	83.28	77.92	8	83.66	78.19	7	83.83	78.23
0.0007	11	82.61	77.90	10	83.32	78.32	9	83.72	78.60	8	83.90	78.67
0.0005	12	82.66	78.18	11	83.36	78.60	10	83.77	78.86	9	83.96	78.95
0.00035	13	82.69	78.37	12	83.40	78.81	11	83.83	79.08	10	84.00	79.14
0.00025	14	82.71	78.51	13	83.43	78.94	12	83.86	79.23	11	84.06	79.27
0.000175	15	82.73	78.60	14	83.46	79.04	13	83.90	79.32	12	84.08	79.39
0.000125	16	82.75	78.66	15	83.48	79.11	14	83.92	79.38	13	84.10	79.44

Table I.1: Frame accuracy for MLP retrain 11