Babelon

Annual Report

Prepared by:

Brno University of Technology

Martin Karafiat, Mirko Hannemann, Karel Vesely, Igor Szoke, Lucas Ondel, Frantisek Grezl, Honza Cernocky

Flexible and semi-automatic system preparation

Based on the recipes „swbd/s5“ for Switchboard in the Kaldi toolkit [Povey et al.(2011)], a set of scripts was created, that should automate the data preparation for a new language pack as much as possible and make it flexible enough to cover all possible Babel languages. We create one configuration file for each language, with such file, it should be possible to process the language pack mostly automatically.

Given that the data format of the Babel language packs is highly standardized, the audio data and the transcriptions can be processed automatically, checking the correct character set and encoding and producing a set of statistics about the distribution of speakers and phenomena found (amount of fragments, noises, foreign speech tags etc.), which helps identifying whether new data fits the assumptions made with Babel data seen so far.

According to the use of the transcriptions as either acoustic model training text (defining also a „clean“ version), language model training text and for scoring purposes, we define different filters, that decide what to do in case of the different annotation tags provided by Appen. For example, we decided to throw away utterances containing speaker changes and replace unintelligible speech with <unknown>.

Most of the manual work is needed on the dictionary and phoneme set. Given the language pack documentation, we generate a list of allowed phonemes and features. During the first year, we encountered features such as syllable stress, syllable tone, romanized transcriptions, upper/lower case, word and syllable (character) based languages.

We encode those features by generating different versions of a phoneme (position dependent phones, stressed/unstressed, different tones). Kaldi has a flexible concept of phonetic context clustering, which does not require to specify phonetic features, but is able to automatically cluster contexts based on similarity. However, extra questions for the decision tree can be given. We automatically generate sets of questions concerning tone, stress and word/syllable position markers and start clustering by putting all versions of a phoneme into one root. To reach optimal performance for each language, different decisions need to be made regarding casing, encoding of stress, tone etc. We always generate a few dictionaries in parallel, as e.g. our HTK/STK system doesn’t work with position-dependent phonemes.

Given a set of phonemes, we identify phonemes, that have poor coverage in the data. The clustering procedure is able to take care of different versions of phonemes that have poor context coverage, while the root phoneme is covered well. However if the root phoneme has poor coverage, phonemes have to be merged. Since the Babel dictionaries encode a lot of variability among dialects, usually these are candidates for merging. Also, phoneme features can be merged (minor and major stress, merging tones etc.) In the future, we could further automate this process by automatically identifying dialect variations.

Automatic analysis of transcription quality

For Vietnamese, we conducted an automatic analysis of the transcription quality, since we found errors in the transcriptions provided by Appen. We expect data of even worse quality in the future BABEL years, so a means to detect transcription errors would be helpful. As a measure of confidence, we compute the expected WER by measuring oracle error rates on lattices decoded on the training data against the reference transcriptions. To investigate the effect on system performance, we trained systems on the parts of the data which have less than a certain oracle WER (Vietnamese):

Simple PLP system on all data	72.5% WER
<30% WER - 1h removed	72.3% WER
<20% WER - 4h removed	72.4% WER
<10% WER - 16h removed	72.5% WER

We observe that the oracle WERs on the training data are quite high. A native Vietnamese speaker listened to few of the utterances with higher oracle WER and confirmed that there are transcription errors. However, surprisingly, this doesn't seem to pose a big problem for maximum likelihood training of systems (just 0.2% WER difference). Nevertheless, for discriminative training techniques, this is different, so that we have to use frame dropping techniques (see below).

Fundamental choices in system building

Using the Switchboard system of the Kaldi speech recognition toolkit [Povey et al.(2011)] as a starting point, a large set of preliminary experiments have been made to find the right training scheme and parameters for the new "Babel-type" data. We summarize some insights below.

To quickly make a number of choices in system building, usually a quick measure – word error rate at a certain system stage - is taken as a criterion. E.g. when choosing the best set of acoustic features, we might use a simple HMM/GMM triphone system. However, our observation is that the decision should be made at the latest possible stage, since speaker adaptive training and discriminative training might change the picture a lot, and we are interested in gains that persist through the more advanced stages of system development. However, longer training times limit the amount of different experiments to be conducted.

More importantly, word error rate might not be a good choice. The Babel data showed, that there can be huge differences in performance among dialects and different evaluation sets, and more importantly, the task at hand is keyword spotting. Given the term weighted value metric, a miss is more severe than a false alarm, and thus a word deletion hurts more than an insertion. NIST tries to account for the different importance of hesitations as errors by making them optionally deleteable. However, an insertion of a hesitation is punished, even if it doesn't harm the keyword spotting performance at all. Hesitations are quite likely to be not annotated as precisely as words, so an insertion should not be punished. Our suggestion is therefore to either always make decisions based on the keyword spotting performance, or if this is too costly, to approximate it by the maximum "word recall" of the speech recognition. Word recall is one minus substitutions and deletions, and we take the maximum out of a set of reasonable operating points (e.g. not going to infinite insertions).

@@@ equation needed

The transcriptions provided by Appen have the special property that they contain a significant portion of unintelligible or foreign speech, cross talk and similar phenomena, which we could summarize as untranscribed or partly transcribed speech. In a first attempt, we filtered all those utterances out to have a cleaner training set to obtain cleaner acoustic models.

In a second strategy, we label them as "unknown speech" and use a garbage acoustic model to align those utterances. Usually, we train our system in stages, using more and more data. The best scheme turns out to be roughly (in utterances) 4k-10k-30k-all. We experimented with different training strategies with this "unknown" data – starting with a clean set training the first stages and adding the "unknown" data in the last stages. However for simplicity, we adapted the scheme of always using it in all stages, since it doesn't seem to harm the performance. For all the languages, a gain of 0.5-1.0% of word accuracy was observed, when including them into the training.

Another peculiarity are the utterances that were prompted ("scripted" data). While the recording condition might fit the conversational data, the language used by the callers definitely does not. So it would be better to leave this data out from language model training, since the evaluation (and most probably also future deployment) of systems is always done on the conversational data. We observed a better perplexity when leaving it out, but it didn't matter for word accuracy. As expected, including the scripted data for training the acoustic models gives usually a (small) gain (see results on NN training for STK system).

Data sets	Cantonese CER (%)	Turkish WER (%)	Pashto WER (%)	Turkish WER (%)
Conversational	47.5	55.9	55.2	57.5
Conversational+ scripted	47.3	54.6	54.8	57.0

Table 4: Effect of using a scripted data in NN training, STK system.

@@@ add title of this section, or, better, move it to the beginning – see general comments

The resulting recipe is to train an HMM-based (Kaldi "tri3b") system with cross-word tied-state triphones, having approximately 4500 tied states and 21 Gaussian mixtures per state, trained from scratch using mix-up maximum likelihood training. In case of the Limited-LP condition we used 2300 ties states and 10 Gaussian mixtures per state. In the next stage (Kaldi "tri4b"), we splice 9 frames and estimate LDA+MLLT and in a final speaker adaptive stage (Kaldi "tri5b") we estimate per speaker fMLLR (40 dimensional). We tuned the parameters (number of Gaussians and tied states, silence modeling, etc.) on the release A held-out sets.

Modeling silence, noise and „unknown"

One problem that caught our particular attention was the huge dependence of performance on the amount and the way of modeling silence (making things wrong or right can make difference of as much as 10% absolute WER). Due to the annotation scheme of Babel that has two parallel channels for each speaker and does only mark the starting point of speech, but not its end (and doesn't guarantee the synchronicity of channels), more than 50% of the recordings contain silence. For Cantonese, this was up to 70%. The transcripts do annotate several kinds of background and foreground noises. In acoustic model training we can either introduce several noise models for each kind of annotated noise, or we can decide to model everything with the silence model.

However, the transcripts also contain a lot of <no-speech> tokens, which can contain silence, but also an unspecified amount of noises and hesitations. In general, the noises are not annotated consistently (understandable, since it is not the goal of the project). In training, the <no-speech> tokens are usually replaced by silence tags. In any case, it leads to polluted silence models.

In HTK, usually the amount of mixtures for the silence model is explicitly tuned and a three-state silence model is used, where the middle state can occur between words (short pause). Kaldi doesn't use a short pause model and the best choice was to use a silence model of six ergodically interconnected states. The amount of mixture components for each model is choosen by the amount of training data (raised by a tunable value „power“) – the Kaldi parameter „power“ had to be increased (from 0.2 to around 0.5) and the „boost-silence“ parameter was used to increase the weight of silence in computing the accumulators.

There are eight types of noises/silence in the transcriptions. Our initial attempt to train eight models performed poorly, due to the polluted models. The next idea was to use three classes of silence models (simple silence, non-speaker noise and speaker noises – also used for the <unknown>) or to have all silences/noises in one class and have one extra model for <unknown>. However, both approaches didn't perform well.

The kind of modeling used for silence had also other interesting side-effects – for example, we observed that subspace Gaussian mixtures (SGMM) models performed particularly worse when using the „wrong“ silence modeling. The most simple and best performing solution so far was to simply put all silences/noises into one model (one line in Kaldi roots file and option „not split“ to disable further clustering) and use this model also to model the <unknown> data (non-speech, unintelligible, foreign, etc.) This is contrary to what we think should be the right way of modeling <unknown> - rather a garbage loop or sub-word model should be used. Therefore, we want to investigate into this further in the future.

Silence reduction in NN training

We found that the Babel data contained huge amount silence (more than 50% in Cantonese). Therefore, we hypothesized that NNs have been focusing too much in silence/speech recognition rather than phoneme-state classification. After removing silence, huge drop of frame accuracy (from 70% to 40%) was observed on cross-validation set during BN-NN training (due to removal of "easy" silence frames) but the final BN features gave us 3.2% absolute improvement, see table 2. The influence (or rather lack of influence) of silence removal is even more interesting with SBN architecture, no drop-off accuracy is observed due to training on huge amount of silence: the first NN is obviously affected by silence but the second one reads already compressed information, therefore it can be better trained.

Features extraction	Cantonese CER (%)

BN – NoSilence Reduction (5Layers)	63.8
BN – Silence Reduction (5Layers)	60.6
SBN – NoSilence Reduction (6Layers)	53.3
SBN – Silence Reduction (6Layers)	53.3

Table 2: Effect (and lack of effect) of silence reduction on various NN architectures for the Cantonese FullLP condition.

Voice activity detection and segmentation

To control the amount of silence (since for evaluation data the segmentation is not given), we trained a Voice activity detection (VAD) model to obtain automatic segmentation. One part that is very sensitive to the correct segmentation is the cepstral mean and variance normalization (CMN/CVN). The mean and variance should be estimated on speech-only portions and with the original segmentation as provided by Appen, the normalization fails. So we use two kinds of segmentations – one directly taken from the VAD and the other one is extended by several frames of silence (usually 15/30) to provide the acoustic model with context.

For VAD, we use a neural network, which gets 465-dimensional features - 15 log-Mel-filterbank outputs, per-utterance mean-normalized, spliced 61-frame context, applying a Hamming window and DCT to obtain 31 bases. We use a 2-hidden layer MLP with 200 neurons in the hidden layers (133k trainable parameters, hidden units are sigmoidal, thesoftmax has two outputs). The MLP was trained for each language on the conversational training part. The MLP posteriors are post-processed by a Viterbi decoder to obtain a more smooth segmentation.

We show the effect of re-segmentation on Cantonese Release B using BUT HTK 69-dimensional features:

System	Train segmentation	Test segmentation	CER [%]
Kaldi	Appen	Appen	47.2 (45.9 – no decoding on empty utterances)
Kaldi	BBN resegmentation	BBN VAD	43.0
Kaldi	Appen – cutting silence on edges by BUT VAD	BUT VAD	42.5

System	Cantonese CER (%) Original segm.	Cantonese CER (%) VAD segm.
Original segmentation for HMM training	49.2	48.8
Resegmentation	47.7	47.3

Table 4: Effect of silence reduction in HMM training for the Cantonese FullLP condition (STK system, PLP+SBN+f0 features) .

Feature extraction

So far, the best keyword spotting systems developed are always based on Large Vocabulary Continuous Speech Recognition (LVCSR) front-end. Accuracy of such keyword spotting system correlates with LVCSR because both tasks require high quality acoustic models. In our approach, we decided to incorporate all high-level techniques into feature extraction as it is easy to share them across the teams.

We concentrated on three main topics:

1. Tonal features and Kaldi feature extraction
2. Neural Network (NN) based feature extraction
    - Stacked Bottle Neck architecture
    - Tonal information - we investigated into using fundamental frequency (f0) and also probability of voicing as additional features processed by NN.
    - Multilingual NN and adaptation into LimitedLP.
3. Discriminatively trained Region-Dependent Transforms (RDT) provided an additional improvement on top of NN based features (that are already discriminatively trained to reduce Frame Error Rate !).

The most important component of our feature extraction is the NN, therefore main experiments for the last year were focused in this domain.

Feature extraction and pitch tracking

Due to the „dominance“ of tonal languages in the first year (Cantonese was released first and Vietnamese was surprise evaluation language) it was natural to investigate into the use of f0 features. As we found later, those features even help for non-tonal languages, so one outcome from the Babel project is that in order to make system building more robust to new languages, we include pitch extraction into the standard set of features used for **every language**.

However, pitch features are very different in nature from the standard frame-by-frame spectral features (PLP, MFCC, filterbanks) – they are not very Gaussian and not even continuous. In a first set of experiments we used the F0 estimation following [Talkin(1995)] and found that the best way is to divide the per-frame F0 by the average speaker F0 to have numbers oscillating around 1.0.

We compared F0 and log F0 features on our STK/HTK system using PLP/VTLN features, normalized using VAD and speaker normalized f0 also taking deltas and double deltas of F0. Results are shown as CER [%] on the heldout set, decoding using the Appen segmentation and training a simple ML system on Cantonese release B:

plp vtln VAD(CMN/CVN)	62.8%
plp vtln VAD(CMN/CVN) + f0log_SpkrNorm	62.0%
plp vtln VAD (CMN/CVN)+ f0log_SpkrNorm (then CMN/CVN)	62.7%
plp vtln VAD (CMN/CVN)+ f0_SpkrNorm	58.6%
plp vtln VAD (CMN/CVN)+ f0_SpkrNorm (then CMN/CVN)	59.0%

Our finding is that plain F0 is giving significantly better results than log(F0), and that it is not beneficial to apply CMN/CVN on normalized F0 features (divided by speaker average f0), it is actually hurting. So we need to switch off CMN/CVN on PLP+NN+F0 feature streams.

We also started using probability of voicing (given by autocorrelation algorithm) as a feature – either concatenated to PLP features (13-dimensional) and also as input to our NN feature estimators. We show results on three Babel languages, using a smaller NN (2M parameters instead of 6M) with a linear bottle-neck layer. We observed, that F0 and probability of voicing is helping for all languages as an additional features for NN training, even for non-tonal languages (training ML system on bottle-neck features):

Features extraction	Cantonese CER (%)	Pashto WER (%)	Tagalog WER (%)
SBN	63.8	57.4	58.6
SBN + f0	50.0	56.4	57.9
SBN + f0 + m	50.0	56.0	57.5

Table 3: Effect of fundamental frequency and probability of voicing on 3 languages, FullLP condition.

To make the features more Gaussian, we apply the log-it transformation to the probability of voicing. A problem with the F0-features is that for unvoiced parts, those features have gaps – a property not well perceived by the Gaussian models. In those regions, usually a default value is applied, however this might cause severe problems during the variance estimation (constant value). Therefore, variance flooring and some kind of noising/dithering must be applied for F0 features.

In the Kaldi systems, we interpolate the normalized f0 features (linear interpolation between gaps, but we want to use better interpolations in the future). On a SAT+LDA+MLLT system on Cantonese (simple ML system, no bottle-necks), there was an improvement from 54.6% to 53.4% CER, and later on, we were able to use CMN/CVN on top of these features, so no special treatment of the normalized features is necessary.

Our final feature set are PLPs (13 dimensions) concatenated with fundamental frequency (F0) and probability of voicing (log-it transformation). Missing F0 values were linearly interpolated and F0 was normalized by division with F0 average over speaker. All features (15 dim) were normalized by CMN/CVN and then either deltas/double deltas were applied or we splice 9 frames and estimate LDA+MLLT and in a final speaker adaptive stage we estimate per speaker fMLLR (40 dim).

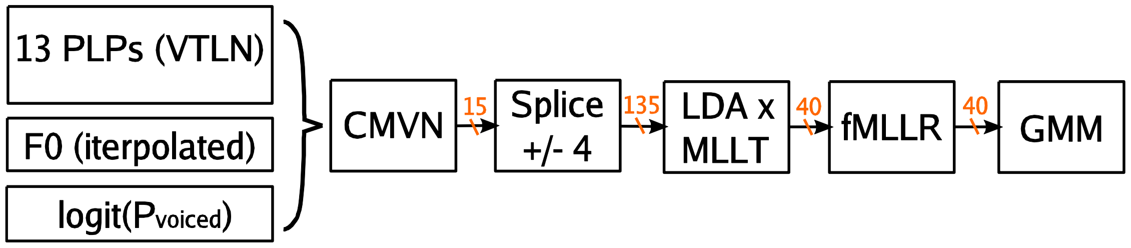

Fig.1 : Feature extraction pipeline for Kaldi GMM systems

Our original VTLN warping factors were estimated by a "fastVTLN" technique, which is based on a codebook of models trained on Switchboard. We found out that these warping factors are not well matched with the Babel data, and so we resorted to the VTLN factors as provided by BBN (estimated on target Babel data only). An experiment shows results using a simple ML Kaldi system using PLP on Cantonese:

- Pure PLP features: 64.2% CER
- VTLN (estimated by SWB models): 63.8% CER
- VTLN (taking BBN warping factors): 63.0% CER

When training a full system (speaker adapted features and DNN), the gain of using VTLN might be smaller, but we still observed almost 0.5% absolute WER difference.

Stacked Bottle-Neck feature extraction

Neural networks were used to generate Bottle-Neck (BN) or Stacked Bottle-Neck (SBN) features. We introduced the SBN structure in [Grezl et al.(2009)] . It contains two NNs: the BN outputs from the first one are stacked, down-sampled, and taken as an input vector for the second NN. This second NN has again a BN layer, of which the outputs are taken as input features for GMM/HMM recognition system. Our previous study [Vesely(2011a)] has shown that BN neurons with linear activation functions provide better performance.

Mel filter-bank outputs were preprocessed for NN training according to Figure 1. We used 15 Critical-Band Energies (CRBE), with conversation-side-based mean subtraction applied. 11 frames of these features are stacked and a Hamming window multiplies the time evolution of each parameter. Finally, DCT is applied, of which 0th to 5th coefficients are retained.

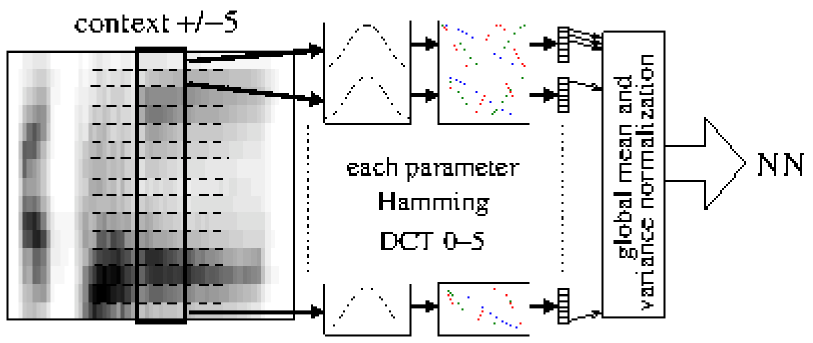

Fig.2. NN preprocessing @@@vector, not bit-map figure!

Stacked Bottle-Neck architecture involoves two NNs: the BN outputs from the first one are stacked, down-sampled, and taken as an input vector for the second NN. This second NN has again a BN layer, of which the outputs are taken as input features for GMM/HMM recognition system.

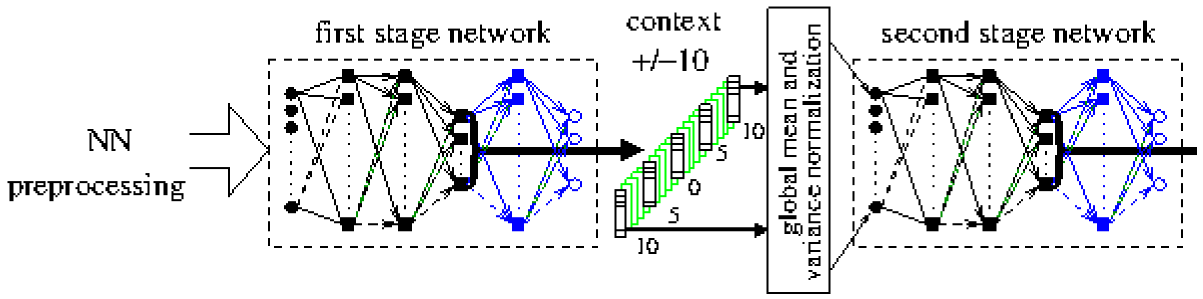

Fig.3. Stacked Bottle-Neck Neural Network @@@vector, not bit-map figure!

Experiments with Bottle-Neck feature extraction

NN Architecture

In our experiments done mostly on Cantonese FullLP, we have found that Stacked Bottle-Neck NN significantly outperforms single BN neural network, see table 1. Moreover, we have found nice (1.9%) improvement by using one more NN layer with fixed number of parameters.

Features extraction	Cantonese CER (%)
PLP	62.8

BN(5Layers)	63.8
SBN(5Layers)	55.2
SBN(6Layers)	53.3
SBN(7Layers)	53.7

Table 1: Comparison of Various NN architectures for the Cantonese FullLP TAR condition.

Unsupervised training for LimitedLP condition

High error rate in Limited Language Pack (LLP) and opening “invisible” part of Full Language Pack for unsupervised training led to experiments with using 1-best transcriptions for NN training. In contrast to HMM/GMMs where transcription errors have significant impact on accuracy, the training of Bottle-Neck feature extractor is less sensitive to transcription errors. This is because the Bottle-neck network performs discriminative compression of input information, which is harder to be biased by a small amount of incorrect training data.. Table 5. shows the dependency of WER on amount of unsupervised transcripts, those were sorted according to a Confidence Measure. The HMM based system was trained on LLP data in all cases, therefore, the results are showing pure effect of NN trained on more data. The table also shows that Bottle-Neck feature extraction has no dependence on Confidence Measures, the best results are obtained by using all the data (+3.6% on Turkish, +4.6% on Pashto). On the other hand, by using the reference (correct) transcriptions (FLP only), we would get almost 4% of further improvement.

Finally, we also made experiments with balancing of training data. The true transcripts were cloned to reach same amount of data size like unsupervised transcripts. Afterwards, the data were shuffled. This procedure gave about 0.5% improvement on both data sets.

Confidence	Turkish Data Size (h)	Turkish WER (%)	Pashto data size (h)	Pashto WER (%)
FLP only	56.7	61.9	64.7	62.0
LLP only	7.38	69.5	7.2	70.2
0.8	+10	67.6	+2.5	68.7
0.5	+29.7	66.3	+15.9	66.6
0.3	+36.8	65.9	+29.1	66.0
0.1	-	-	+47.3	65.7

0.0	-	-	+58.6	65.6

Table 5: Effect of adding unsupervised transcriptions into Limited Language Pack training.

Multilingual SBN adaptation into Limited LP

@@@sem pridat poznamku, ze toto uz je jina podminka, protoze pouzivame data z jinych jazyku

In previous work [Vesely(2012)], we trained multilingual bottle-neck NN. As can be seen in Fig 4., in this type of network all the hidden layers are language-independent, while the output layer is divided into the language-dependent submatrices with its associated softmaxes. Due to the structure of the model, the weights located in hidden layers compute features which generalize across languages, because only the output layer contains language-specific weights. We will call this approach by "block softmax".

These observations led us to consider the adaptation of NN trained on non-target Babel languages to target Limited Language Pack domain. The proposed approach should allow us to train a larger network, because the parameters can be estimated on larger amount of training data. This network will serve as the starting point for training a NN on the target language, where the amount of the data is small. The retraining will be fast even for large NNs, which is a second strength of the presented approach.

During our various experiments, we have observed that this "block softmax" approach has sometimes problem with training convergence and requires a reduction of learning rate. To prevent this behavior, we first trained an NN structure with just "one softmax" for all languages and used this NN as initial weights for "init block softmax" training. This initialization was done only for the first stage NN.

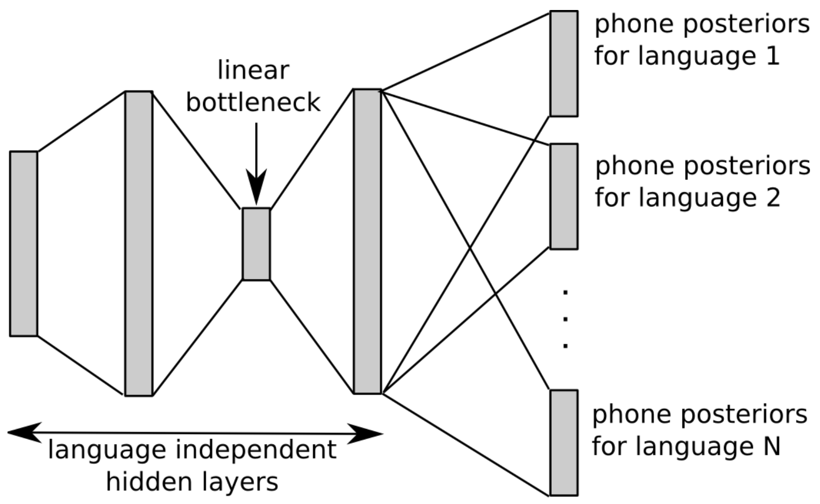

Fig 4. Multilingual bottleneck network

The ultimate goal was to do the adaptation of the "Convolutive Bottleneck Network" [Vesely(2011b)], where all the weights are trained at once. As can be seen in Fig 5., in this approach, the context expansion which is done between the two networks is moved in front of the whole structure, the first network is cloned having shared weights, while the global normalization between the two networks is omitted. The first stage NN is present in the convolutive structure five times, but all its instances share the same weights.

The "Convolutive Bottleneck Network" and "block softmax" approaches were used together. The training of this feature extractor was done in three phases:

1. the first stage NN is trained in standard way with "one softmax".
2. the NN is retrained with "block softmax" -> "init block softmax".
3. the whole structure is created with the random initialization of the second stage NN and trained

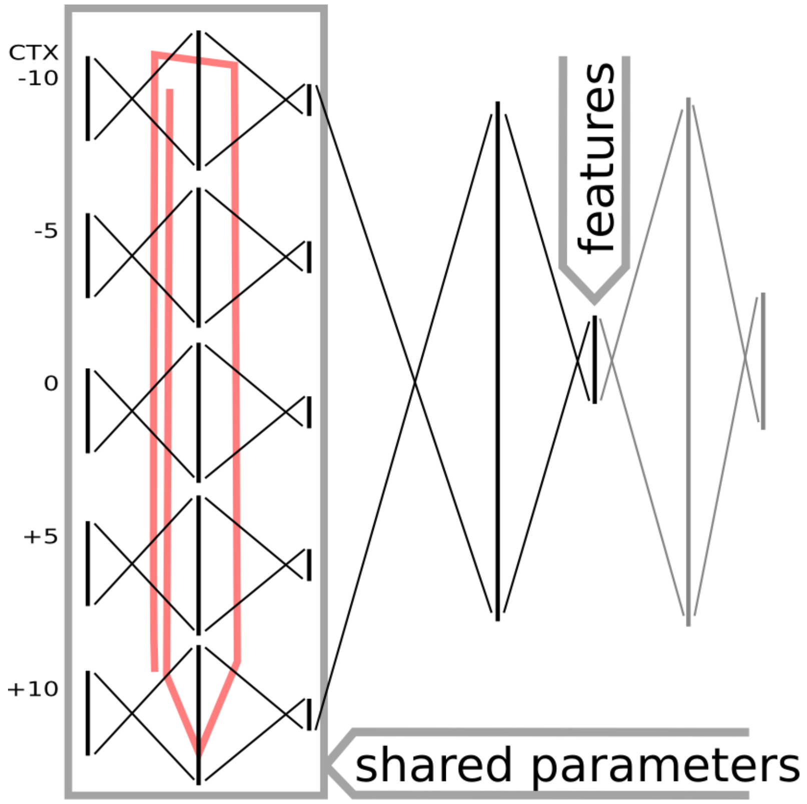

Fig 5. Convolutive Bottleneck Nework scheme

The adaptation of NN is done through retraining of an already trained NN with smaller learning rate on target data. Such pre-trained NN provides us a good starting point which already produces reasonable features. Retraining with small learning rate allows to slightly steer the weights towards the acoustic space of the target data.

The two most promising multilingual approaches "init block softmax" and "Convolutive block softmax" were used for adaptation on all languages. The results are shown in Tab.6. The NN was trained on 3 Full Language Packs and adapted into target Limited Language Pack.

When comparing the two approaches themselves, the results are very similar except Cantonese, where the difference is more than 1%. Thus it can be said that even the simpler adaptation technique, which does not employ the computationally costly Convolutive NN, performs very well. Training Convolutive NN brings only a very small additional improvement. The difference of 0.1% for Tagalog is negligible.

Comparison with the baselines shows that with the adaptation, we reach reach about half of the gap between BN features trained on LLP and FLP. The improvement is about 4% absolute in all cases.

System	Cantonese CER (%)	Pashto WER(%)	Turkish WER(%)	Tagalog WER(%)
Monolingual FLP	56.1	62.0	61.9	63.6
Monolingual LLP	64.8	70.2	69.5	70.8
Init block softmax	61.3	66.0	65.9	66.9
Convolutive block softmax	60.1	65.5	65.6	67.0

Table 6: LLP systems with BN features obtained from adapted multilingual NNs.

For Surprise Language Evalutation, we trained initial NN on all 4 FullLP and adapted into Vietnamese LimitedLP. The results are shown in tab.7

System	Vietnamese WER (%)	Vietnames WER(%) NN trained also on balanced unsupervised transcripts
Monolingual LimitedLP	73.0	67.0
Init block softmax	67.7	66.3
Convolutive block softmax	68.1	66.5

Table 7: Vietnamse LLP system with BN features obtained from adapted multilingual NNs.

Further improvement given by re-aligning and larger amount of NN training data

The temporal alignment of NN training targets is given by forced alignment with a baseline PLP system. But as the system performance significantly increases by employing the new NN features, it makes sense to rebuild the alignments, and re-train the whole network.

Also, originally we omitted sentences which contain the "<unk>" symbol from the training , this was to make sure that we are training on intelligible speech only. Instead of excluding the data, we introduced a new phoneme "<unk>", and mapped the problematic data on it.

Table 8 presents the effect of NN target realignment and addition of more training data (containing "<unk>") on Vietnamese Limited LP.

System	Vietnamese CER (%)
LLP	73.0
+ Realignment	71.5
+ Realign + UNK	70.6

Table 8: Effect of realigning of NN targets and<unk> segments on the Vietnamese Limited LP condition.

Deep Neural Networks and Deep Belief Networks

Recently, Deep Neural Networks (DNN) and Deep Belief Networks (DBN) have been very successfully used as acoustic models. Also our experiments on neural network training suggest the advantage of using deeper nets and a linear bottle-neck layer. So for our Kaldi systems, we trained a DNN/DBN as acoustic model and used its outputs (state posteriors) directly in decoding – this is called „hybrid“ approach, which uses the state posteriors as inputs to the HMM.

The DNN is trained optimizing frame-level cross-entropy using mini-batch Stochastic Gradient Descent. The targets for training are obtained from the state alignments using the speaker adapted HMM/GMM system (Kaldi system „tri5b“). Targets are the triphone-tied-states obtained from triphone decision tree clustering - 4.5k outputs – which is the number of GMM PDFs. On the input to the network we splice 11 frames of fMLLR features (440 dimensional), and shift and scale the features to have zero mean and unit variance. The fMLLR features were described above. @@@Martas needs to add these

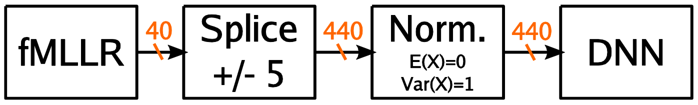

Fig 6. Input features for Kaldi DNN/DBN systems

In order to avoid over-fitting, we monitor the frame accuracy on the cross-validation set (10% of training data), and apply learning rate halving, when the per-epoch improvements get lower than 0.5%. Once the halving started, we always halve until the per-epoch improvement gets lower than 0.1%. This is similar to the procedure described in [Hinton et al.(2012)]

The next step was to train the DNN in a sequence discriminative manner – by further training four epochs using the MMI criterion. The DNN is re-trained optimizing the (MMI/sMBR) criterion, which is done by Stochastic Gradient Descent with per-utterance updates [Vesely et al.(2013)]. We used the development set to check which iteration gave the best WER.

The Viterbi alignment to the reference is used as numerator lattice and for the denominator we use a lattice produced with a weak language model (unigram). Both lattices are generated with the cross-entropy based DNN system. The gradient is back-propagated into the network: it is the posterior from the Forward-Backward algorithm over the denominator lattice minus the correct path from the numerator.

We show a set of experiments for Cantonese with a hybrid system using speaker adapted (fMLLR) features (PLP+F0) as described above:

Baseline: SAT system on PLP(VTLN)+F0(interp.)+voicing:	*52.1% CER*
DNN on PLP(VTLN)+F0(interp.)+voicing:	*45.7% CER*
DNN-MMI sequence training on top of that:	*43.6% CER*

Further improvement can be obtained by using unsupervised pre-training. We pre-train (unsupervised) a DNN layer-wise by stacking 6 RBMs (radial basis function model), using the contrastive divergence optimization and after pre-training we add the final layer of the DNN and re-train the whole network by optimizing cross-entropy. Experimental results are given for Turkish FullLP, where we also show unnormalized MTWV for the PI condition (no added keywords):

Baseline: SAT system on same features: 57.4% WER

System	WER	MTWV
(A) DNN	50.0%	0.5170
(B) DNN MMI	47.5%	0.5254
(C) DBN	48.2%	0.5446
(D) DBN MMI	45.9%	0.5873

We see, that the DBN with pre-training have better performance. We observed that DNN/DBN produce a significant improvement of keyword spotting performance. We assume that deeper NN produce „sharper“ posteriors. In fact the improvement is caused by pre-training, deeper structure (4->6 hidden layers) and larger hidden layers (1024->2048 units).

On Vietnamese we reached a further improvement of 0.5% WER by adding another cross-entropy iteration after re-alignment and by performing cross adaptation. Here, we use the state Minimum Bayes Risk (sMBR) criterion instead of the MMI criterion. Also in the sequence based training, the lattices are re-generated after the first iteration:

59.2% GMM baseline (F0, VTLN, re-segmentation, LDA+MLLT+SAT)

51.7% DNN (RBM pre-training, 6 layers, 2048 sigmoid units, per-frame cross entropy training)

51.0% DNN (re-aligned by DNN)

47.6% DNN - RBM->Xent->sMBR->lat->sMBR

47.1% DNN - RBM->Xent->*Xent*->sMBR->lat->sMBR->*CrossAdapt*

unnormalized MTWV: 0.3976 normalized MTWV: 0.5276

@@@not sure for which system you give MTWVs.

@@@if you have them, add also MTWVs for the baseline GMM system

Unsupervised training of DNN and confidence estimation

Inspired by unsupervised experiments from [Novotney et al.(2009)], we tried to apply a similar approach to DNN training. For the LimitedLP condition, we take the 10h of transcribed speech for supervised training and use the remaining ~90h of speech for unsupervised training. Experimental results are given for Vietnamese. In this scenario we first built a supervised DNN system (including sMBR training). The procedure is very similar to the FullLP system, but here we used a smaller network (1024-dimensional hidden layers, 2174 outputs). The seeding system has a WER of 61.0% (GMM baseline ~70% WER).

With this DNN system we decoded the unsupervised training data. Due to the high error rates, it is necessary to select utterances for training, wich are likely to be correct. We produced both per-utterance and per-frame confidence measures. For the utterances we used average per-word Minimum Bayes Risk, and as per-frame confidence we used the forward-backward posterior under the best path in the lattice, that we computed in the ’PDF-state’ level.

As a side experiment, we also decoded the supervised training set and removed 1% of the training segments according to the confidences (without using the reference transcriptions). By training on 99% of the training segments we obtained a WER of 60.7%.

In order to find the optimal mix of the training data, we ran several quick sets of experiments, using a smaller DNN without pre-training (4 hidden layers, 1024 units each, also using a different VAD and dictionary, so numbers are not comparable with other sections). At first, we added unsupervised segments sorted by confidence. The best was to add 80% of the data (WER 63.6% → 61.9%). Then we tested multiplying the occurrences of the supervised data to achieve weighting. The best turned out to be to add the supervised data four times (WER 61.7% → 61.3%).

Finally, we experimented with frame dropping based on per-frame confidence. Here, we tuned the confidence threshold, which rejects non-confident frames. The optimal value was 0.9, meaning that only very confident frames were accepted and in total, 21% of training frames were rejected (WER 61.3% → 60.9%). The total absolute WER improvement from unsupervised Cross-entropy training was 2.7%.

At this point we kept the hyper-parameters to prepare the mix of training data, and we trained a “full” DNN (RBM pre-training, 6 hidden layers, 1024 neurons), leading to a WER improvement of 63.7% → 61.8% at the cross-entropy level. However, on the sMBR level (sMBR with 10h) the WER improvement was only 61.0% → 60.0%, so we decided to try unsupervised sMBR training as well.

The unsupervised sMBR was done with the same mix of data as in the Cross-entropy training (80% of unsupervised data, 4x supervised data). Due to lack of time, we re-used the seeding-system lattices and 1-best paths for the unsupervised data (on the supervised 10h part, we generated new lattices and alignments). The optimal frame-dropping threshold was 0.25, meaning that almost all the unsupervised frames were beneficial (only 1.5% dropped). The WER improvement from unsupervised sMBR training was 60.0% → 59.6%, which is the best score of the Kaldi LimitedLP system.

The total improvement with respect to the seed system is 61.0% → 59.6%, which shows that we should focus more on this scenario in the future, there should be space for further improvement. (Note the WERs are with %hesitation scored, the final WER w/o hesitation is 58.1%.)

Fusion of Lattices for keyword spotting

BUT has two good ASR Kaldi systems using two different acoustic models: DNN and GMM (on top of Bottleneck-RDLT-features). Naturally, we wanted to fuse them in a way that is beneficial for both STT and KWS. Inspired by a fusion example in Kaldi, we adopted the already implemented lattice combination and Minimum Bayes Risk decoding [Swietojanski et al.(2013)].

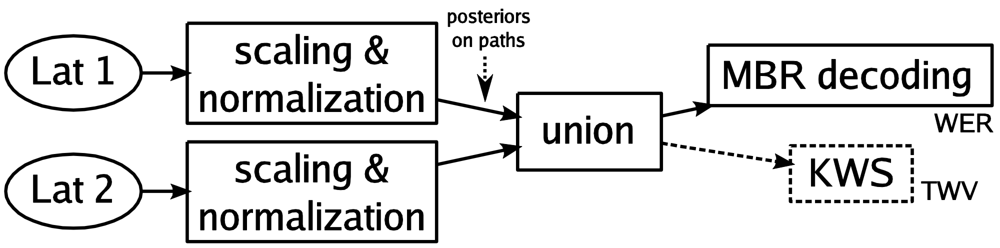

Fig. x Lattice fusion scheme used for decoding and keyword spotting

During lattice combination, each source lattice is re-scaled by an acoustic weight and normalized by removing the total cost of all paths (backward cost), finally the lattices are re-weighted and unioned (FST operation). This procedure leads to lattices which already contain optimally scaled AM/LM scores, and can be used either to decode the best hypothesis (using Minimum Bayes Risk decoding), or to perform KWS.

The experimental results on Vietnamese:

- Train GMM system on top of BUTv2 features: 46.5% WER
- DNN system: 45.5% WER, 0.57 normalized MTWV
- lattice combination: 43.6% WER, 0.59 normalized MTWV

Grapheme based recognizers

One of the most cost labor intensive steps in preparing a speech recognizer for a new language is the creation of a pronunciation dictionary together with designing a set of acoustic units (phonemes) to be used for recognition. For all languages for which the spelling is closely related to the pronunciation, it should be possible to build grapheme-based dictionaries by mapping each letter in the alphabet to a „graphone“ unit. With the help context-dependent graphones, it should be possible to use them as acoustic modeling units. In such case, (almost) no effort is needed to create dictionary and phoneme set.

For a simple ML-system for Turkish FullLP using the BUT 69-dimensional features (PLP HLDA, stacked bottleneck, F0+delta+ddelta), we obtain even better performance for the grapheme based system than when using the dictionary provided by Appen, which indicates that the Turkish pronunciation is highly regular (and the supplied pronunciation dictionary might contain significant portion of errors):

Language/System	**Turkish**	**Pashto**	**Tagalog**	**Vietnamese**
phoneme based system:	49.1% WER	56.5% WER	56.6% WER	47.9% WER
grapheme based system:	48.6% WER	56.7% WER	59.7% WER	48.4% WER

Table x: @@@add title

For Tagalog and Vietnamese, the situation is a bit different, the grapheme based system performs worse. The reason here is that, especially for Tagalog, there is a high amount of foreign words present (Tagalog contains many words of Spanish origin and many English words with English pronunciation, in Vietnamese are many English, some French words). The spelling/pronunciations of the foreign/loan words follow different patterns than the native words.

A strategy here is to automatically identify all words which are not native (see work done by NWU) and to train the acoustic models only on the native words, or to devise a different set of graphones for the foreign words. This has to be investigated further in the future.

Even if the grapheme based system is not outperforming the phoneme dictionary based system, having two different sets of acoustic models is still valuable, since we can do a fusion of the two systems by either ROVER/consensus techniques or using one of the systems as a first pass and doing cross-adaptation. On Vietnamese limited LP, we observed a 0.2% absolute gain in WER when doing cross-adaptation on the grapheme based system.

BUT Decoding and Keyword spotting

The final word transcriptions are decoded using a 3-gram Language Model (LM) trained only on the transcriptions of training data (conversational+scripted). We use the Kaldi recipe for the generation of the decoding network (using a log-semiring finite state transducer compiled from language model, dictionary and triphones to Gaussians mapping close to [Mohri et al.(2008)]) We generate word lattices in Kaldi-WFST format [Povey et al.(2012)] with the decoding network using the final features. We chose the best acoustic-scale / LM-scale according to the dev set, we also made sure that the pruning beams are chosen appropriately. One insight from the keyword spotting experiments was, that lattices should be extraordinarily huge to achieve the full performance (used Kaldi beam of 18.0 and lattice-beam 10.0).

The search was done using the Latt2Multigram tool (http://speech.fit.vutbr.cz/software/lattice-spoken-term-detection-toolkit-latticestd [Szöke(2010)]). The terms are represented by sequences of labels (which must correspond to the labels in the lattices). The tool searches for the exact path in the lattices representing the term (one or more consecutive labels). When the path is found, the posterior probability is estimated in forward/backward ”Baum-Welch” style. In case of several overlapping detections (at least one frame) of the same term, the posterior probabilities are summed.

All experiments are in the WORD PI condition: We search in word lattices for the word representation of terms (as was provided by NIST). No term processing nor conversion into characters was done. If there is an out-of-vocabulary word in the term, the term can not be found - even in the case of multi-word terms (we require an exact path match in the lattice).

The BUT KWS is based on the following process [@@@cite Igi's paper or thesis]:

1) We expect standard lattices (no confusion networks) as an input. Each link represents lexical unit (word, syllable or phoneme multigram).

2) We process the lattice by !NULL node removal and minimization (using SExpand from STK toolkit).

3) We search for exact match of consecutive links in lattice representing given term (one link for one syllable long term, 5 links for 5 syllable long term for example).

4) Each of such link sequence occurrences in lattice is called TermDetection.

5) Confidence of each TermDetection is posterior probability. It is accumulated likelihood of given link sequence + accumulated forward likelihood (alpha) + accumulated backward likelihood (beta) divided by accumulated forward likelihood of the whole lattice. The accumulation of alpha/beta is Baum-Welsh or log-semiring approach.

6) Overlapped groups of TermDetections (at least one frame overlap) are found. These are called TermGroup

7) Confidence of each TermDetection in the TermGroup is re-calculated as sum of posteriors of overlapped TermDetections - note that not all TermDetection in group must be overlapped with each other. The summation is known as C-max approach.

8) The TermDetection having the best re-calculated confidence is found in the group.

9) This best TermDetection represents the occurrence of the term in the data - so its confidence and its timing.

UBTWV

We used UpperBound-TWV for our internal evaluation of KWS. One feature of NIST TWV metric is its one global threshold for all terms. This is good for evaluation for end-user environment. On the other hand, it leads to uncertainty in comparison of different experimental system setups. We do not know if the difference is caused by different systems or different normalization and global threshold estimation. This is reason for our definition of Upper Bound TWV (UBTWV) [@@@cite Igi's paper or thesis]. The difference to TWV is in individual threshold per each term. The ideal threshold for each term is found to maximize term's TWV.

This is equivalent to shifting the score of each term, so that maximum term's TWV is obtained at threshold 0.0. Two systems can be compared by UBTWV without any influence of normalization and ideal threshold level estimation in systems producing TWV score. The actual and maximal values are equal for UBTWV and both are denoted by UBTWV. In other words, as TWV is pooled metric (you put all term into one pool and have one threshold), the UBTWV is non-pooled version of TWV.

The information we get from UBTWV is the "space left" for calibration. If, for example, we find that non-calibrated systems has TWV 0.4, calibrated TWV 0.5 and UBTWV is 0.6, then we see that the calibration brings about half the difference between TWV and UBTWV.

If we produce a new system (non-calibrated) which has for example TWV 0.35 but UBTWV 0.8, we know that it is good system, but rather badly calibrated. Opposite to a system (non-calibrated) with TWV 0.45 and UBTWV 0.5 where the system itself is well calibrated but there is no much room for improvement by calibration itself.

We found the UBTWV score reliable enough for estimation of TWV of systems calibrated by BBN during the development and evaluation period. This saved some time during the development phase. We had not need to calibrate every system output.

AA condition - adding words from terms into the language model

We experimented with adding words from term-list into the recognition vocabulary and language model:

1) We extracted all possible word/syllable (depends on the word/syllable character of the language) n-grams from particular terms. Then we boost existing n-gram probability in the LM by log-adding with 1 and we add non existing n-grams with singleton probability (boosted by logadd with 1).

2) We found that this approach does not perform consistently (Limited vs. Full set, Dev vs. Eval termlist). It helped mainly on Limited set Turkish LP (about 3 points on TWV), Pashto LP (about 2 points on TWV), Tagalog LP (about 7 points on TWV) and Cantonesse LP (about 4 points on TWV). However, on Full set, there was not significant improvement or small deterioration (Cantonese Kaldi system).

According to this conclusion and the syllable character of Vietnamese data, we decided not to submit Vietnamese AA system.

MPE training

We found, that MPE retraining of STK system brings significantly better word error rate (several percent absolute), has positive impact on generated lattices (MPE lattices have 1/3 size of the MaximumLikelihood lattices), but the impact on keyword spotting accuracy was not significant (from 0 to 1 point of TWV).

Phoneme multigrams

We experimented with subword systems based on word boundary phoneme multigrams [@@@cite Igi's paper or thesis]. The multigrams are derived from phoneme alignment of the training data with marking of word boundary.

The training of the multigrams was done in bottom-up manner and maximum likelihood framework. After the initial collection of all possible multigrams from the data, the "cross-word" multigrams were removed and the inventory was trained. Finally, standard 3-gram language model was estimated on the multigram segmentation of the training data. The length of multigrams was set up to 7 phonemes (incl. the boundary mark).

Terms were converted to phoneme sequences (using dictionary or automatically trained G2P) and then segmented into multigrams.

Multigram systems have some difficulties during lattice generation (slow decoding, huge lattices) and STD (slow keyword search). We had to limit the decoding beam which led to suboptimal performance - there is some room for improvement of these systems, but it takes huge amount of CPUs.

The multigram systems were slightly (0 to 2 points on TWV) worse than the PI word systems on Limited set and about 3 points worse on Full set. However they are good for non-linear combination of word (in-vocabulary terms) and multigram (out-of-vocabulary terms) systems in PI condition. In that case, we obtained improvement. However due to syllable character of Vietnamese, we do not used multigram system in the evaluation. It does not bring any improvement.

Language modeling and automatic data acquisition

Due to the huge success of recurrent neural network language models (RNNLM) [Mikolov et al.(2010)], we also wanted to take advantage of that for the Babel languages. We trained a RNNLM on the training transcriptions and either used it for re-scoring N-best lists generated from the lattices or used the RNNLM to generate additional, artificial training text, train another N-gram on that and interpolate it with the original N-gram model trained on the transcripts.

Another stategy to improve the language model is to automatically acquire new data (from the web). We crafted some scripts that based on the vocabulary in the training lexicon harvest tweets from Twitter containing mostly in-vocabulary words, or made use of the OpenSubtitles library (part of the OPUS project http://opus.lingfil.uu.se/). The text from OpenSubtitles had to be normalized (mostly by expanding digits by the ICU library). A peculiarity are the Cantonese transcriptions. Since these are written in simplified Mandarin characters and use something like a „Mandarin style of writing Cantonese" (according to one Mandarin speaker), they are not (easily) combinable with other Cantonese sources – neither from Hongkong (traditional characters), neither from Chinese Cantonese provinces.

For RNNLM N-best rescoring, we typically observe a gain of 1% absolute WER on the Babel languages. However, these gains cannot be ported to keyword spotting, since the lattices for keyword spotting are quite large, and the number of paths in the lattice is exponentially growing with length of the utterance and dephts of the lattice, so N-best rescoring is insufficient to produce significant amounts of keywords. The second strategy to enhance the n-gram language model by automatically generated text from the RNNLM, gives smaller gains: e.g. 0-0.5% absolute WER. The resulting LMs from generated text are quite huge and need to be pruned, however we are not yet sure about the right pruning technique.

Example on Turkish:

Training data	- 260 kWords
RNN automatically generated data	- 42.5 MWords
OpenSubtitles2011	- 190 MWords (Cleaned, digits expanded using ICU)

@@@say what is ICU – add hyperlink or [citation]

We were able to achieve a perplexity reduction from 321 (train data only) to 276.823 (all). However, this resulted in a WER reduction of 0.2% absolute only – using a dictionary from training data, which resulted in 7.8% OOVs. It is necessary to expand the dictionary by G2P techniques.

Concerning the Twitter data, we performed an experiment on Tagalog, where we interpolated with the original language model by only adding in-vocabulary n-grams. We achieved an 0.4% absolute improvement in WER, however the MTVW stayed basically the same or even slightly deteriorated (0.2811 to 0.2783) To conclude, so far we didn't see big gains from adding additional data or using RNNLMs, but we want to explore other ways to unleash their potential in the future.

1 Babel-related Scholarly Activities including papers published and presentations given during this period

Paper presented at ICASSP 2013 (Vancouver, Canada)

- Hannemann Mirko, Povey Daniel, Zweig Geoffrey: Combining Forward and Backward Search in Decoding, In: Proceedings of ICASSP 2013, Vancouver, CA, IEEESP, 2013, s. 6739-6743, ISBN 978-1-4799-0355-9

Papers accepted for Interspeech 2013 (Lyon, France):

- K. Vesely, A. Ghoshal, L. Burget, D. Povey: Sequence-discriminative training of deep neural networks
- M. Karafiat, F. Grezl, M. Hannemann, K. Vesely, J. Cernocky: BUT BABEL system for spontaneous Cantonese
- F. Grezl, M. Karafiat and K. Vesely: Adaptation of Neural Network Feature Extractor for New Language (to be submitted for ASRU)

2 References

- [Grezl et al.(2009)] F. Grezl, M. Karafiat, and L. Burget, "Investigation into bottle-neck features for meeting speech recognition." in Proc. Interspeech 2009, no. 9, 2009, pp. 2947–2950.
- [Hinton et al.(2012)] G. Hinton, L. Deng, D. Yu, G. Dahl, A. rahman Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition." IEEE Signal Processing Magazine, November 2012.
- [Mohri et al.(2008)] M. Mohri, F. C. N. Pereira, and M. Riley, "Speech recognition with weighted finite-state transducers." in Handbook on Speech Processing and Speech Communication, Part E: Speech recognition, L. Rabiner and F. Juang, Eds. Heidelberg, Germany: Springer-Verlag, 2008, p. 31.

- [Mikolov et al.(2010)] Mikolov Tomáš, Karafiát Martin, Burget Lukáš, Černocký Jan, Khudanpur Sanjeev, “Recurrent neural network based language model” in: Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010), Makuhari, Chiba, JP
- [Novotney et al.(2009)] S. Novotney, R. Schwartz, and J. Ma, “Unsupervised acoustic and language model training with small amounts of labelled data.” in Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, ser. ICASSP ’09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 4297–4300. [Online]. Available: http://dx.doi.org/10.1109/ICASSP.2009.4960579
- [Povey et al.(2011)] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, “The Kaldi speech recognition toolkit.” in Proc. ASRU. IEEE, 2011.
- [Povey et al.(2012)] D. Povey, M. Hannemann, G. Boulianne, L. Burget, A. Ghoshal, M. Janda, M. Karafiat, S. Kombrink, P. Motlicek, Y. Quian, N. Thang Vu, K. Riedhammer, and K. Vesely, “Generating exact lattices in the WFST framework.” in Proc. ICASSP. IEEE, 2012, pp. 4213–4216.
- [Swietojanski et al.(2013)] P. Swietojanski, A. Ghoshal, and S. Renals, “Revisiting hybrid and gmm-hmm system combination techniques.” in Proc. IEEE ICASSP 2013, 2013.
- [Szoke(2010)] I. Szoke, “Hybrid word-subword spoken term detection.” Ph.D. dissertation, 2010.
- [Talkin(1995)] D. Talkin, A robust algorithm for pitch tracking (RAPT). New York: Elsevier, 1995, pp. 495–518.
- [Vesely(2011a)] K.Vesely, M.Karafiat, and F.Grezl, Convolutive bottleneck network features for LVCSR, in Proceedings of ASRU 2011, 2011, pp. 42--47.
- [Vesely(2012)] K.Vesely, M.~Karafiat, F.Grezl, M.Janda, and E.Egorova, The language-independent bottleneck features, in Proceedings of IEEE 2012 Workshop on Spoken Language Technology. IEEE Signal Processing Society, 2012, pp.336--341.
- [Vesely(2011b)] K.Vesely, M.Karafiat, and F.Grezl, Convolutive bottleneck network features for LVCSR, in Proceedings of ASRU 2011. IEEE Signal Processing Society, 2011, pp. 42--47.