Low-Resource Speech Recognition and Keyword-Spotting

M.J.F. Gales, K.M. Knill and A. Ragni

Cambridge University Engineering Department, Trumpington Street Cambridge
{mjfg,kate.knill,ar257}@eng.cam.ac.uk

Abstract. The IARPA Babel program ran from March 2012 to November 2016. The aim of the program was to develop agile and robust speech technology that can be rapidly applied to any human language in order to provide effective search capability on large quantities of real world data. This paper will describe some of the developments in speech recognition and keyword-spotting during the lifetime of the project. Two technical areas will be briefly discussed with a focus on techniques developed at Cambridge University: the application of deep learning for low-resource speech recognition; and efficient approaches for keyword spotting. Finally a brief analysis of the Babel speech language characteristics and language performance will be presented.

1 Introduction

In recent years there has been an increasing interest in Automatic Speech Recognition (ASR) and Key Word Spotting (KWS) for low resource languages. One of the driving forces for this research direction was the IARPA Babel project [13] which ran from March 2012 until November 2016. To quote from the BAA:

> "The Babel Program will develop agile and robust speech recognition technology that can be rapidly applied to any human language in order to provide effective search capability for analysts to efficiently process massive amounts of real-world recorded speech."

The particular form of speech technology assessed as a realisation of this aim was Key Word, or phrase, Spotting (KWS). The funding for, and evaluations of, the project was split into four phases, a base period (BP) followed by three "option" periods (OP1, OP2 and OP3). During the project 25 languages were released spanning a wide range of language groups, writing schemes, and linguistic attributes. Conversational telephone speech data was recorded either directly, or using a microphone. Each side of the conversation was recorded separately.

Language packs were released for each language with various quantities of data:

- **Full Language Pack** (FLP): 40-80 hours of transcribed audio data;
- **Limited Language Pack** (LLP): 10 hours of transcribed audio data, selected from a subset of conversation sides in the FLP;

– **Very Limited Language Pack** (VLLP): 3 hours of transcribed audio data, selected from all sides in the FLP. This was a baseline for active learning approaches.

In addition untranscribed audio data was made available, yielding approximately 100-150 hours of audio data in total per language. For each phase of the programme the evaluation concentrated on different configurations: BP FLP; OP1 LLP; OP2 VLLP; and OP3 FLP. The results presented in this paper are based on the FLP configuration as this was the focus in the final phase of the project.

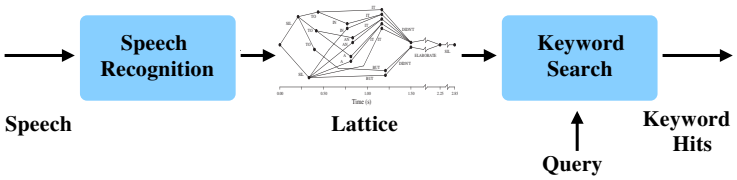

Fig. 1. Key Word, or Phrase, Spotting Pipeline

The vast majority of systems developed followed the pipeline shown in figure 1. An Automatic Speech Recognition (ASR) system is initially run to generate lattices, with nodes at phone, morph or word level. Using lattices to propagate information from the ASR system to the KWS stage makes the system less sensitive to errors; words and phrases can be found even if they do not appear in the 1-best ASR output. Given the quantities of data available in these low resource scenarios the Word Error Rates (WERs) can be very high, 30% to 70%, which means that very rich deep lattices are required for high performance KWS.

Error mitigation between the ASR and search module is only one of the problems that must be dealt with. Current state of the art speech recognition systems are based around deep learning [16]. These approaches operate best when there are large quantities of training data, the opposite of the situation in the Babel project. To address this problem approaches have been developed for both the acoustic and language models used in the majority of ASR systems; examples of these will be discussed in section 2. Another factor that impacts the development of low-resource systems is the lack of linguistic resources. Unlike more frequently investigated languages, there are unlikely to be well defined lexicons, morphological analysers or parts of speech taggers. To address this the impact of using purely graphemic systems [17,18] on a range of languages using both Latin and non-Latin scripts is discussed. In addition, approaches for system combination for both ASR [8,6,27] and KWS [22,30,28,19] and their impact in a low-resource scenario will be described.

One of the interesting aspects of the Babel data is that there are 25 languages with a wide range of attributes, with all the data collected and annotated in a

consistent fashion on a highly challenging task. The final part of this paper discusses the performance over a number of these languages in a consistent configuration.

For the Babel project the performance of the system was evaluated in two ways. The primary metric reflected KWS performance. The Term Weighted Value (TWV) [9] is defined as

$$TWV(\theta) = 1 - [P_{miss}(\theta) + \beta P_{fa}(\theta)] \quad (1)$$

where $P_{miss}(\theta)$ and $P_{fa}(\theta)$ denote the probability of miss and false alarm, respectively, and β is 999.9. For the evaluation a single threshold was required to be specified. To avoid the impact of threshold selection, in this paper the Maximum TWV (MTWV) score is given which is the maximum TWV achievable for that system. The second metric used is related to the WER for ASR. As the specification of a word can be poorly defined for some languages, the metric used is the Token Error Rate (TER). This is defined in the same way as the WER, but with the generalisation of handling tokens rather than words. For most languages TER and WER are the same.

2 Low-Resource Speech Recognition

Speech recognition for low-resource languages has followed the same directions as more general speech technology. Deep learning is a central aspect of all components of these systems: feature extraction [15,12]; acoustic modelling [16]; and language modelling [21]. To address the lack of training data, however, a number of modifications have been made to the standard pipeline. These approaches include data augmentation [23,5,14]; the use of web data [20]; extensive system combination [31]; and the use of multiple languages [4,24]. Furthermore the concept of *low resource* can be applied beyond the availability of training data to include linguistic resources such as an accurate lexicon.

This section briefly describes some of the approaches adopted for implementing the lexicon, acoustic and language models in those low-resource scenarios at CUED to support the evaluation systems developed over the Babel programme [11,26,25]. For the sake of brevity the baseline ASR system will not be described in detail. For details of the systems used see the associated references.

2.1 Graphemic Lexicon

For the OP2 and OP3 evaluations no phonetic lexicon was supplied. To address this a graphemic lexicon was used. Here, the spelling of the word is used to directly determine the sub-word units. Prior to the Babel program, most graphemic systems have been built either for Latin script languages or the script has been converted to Romanised form before creation of the graphemic lexicon, e.g. [17,18]. One of the challenges was to examine whether a graphemic system

could be applied to a wide range of languages with minimal, or preferably no knowledge of phonology of the language.

The approach adopted at CUED to constructing graphemic systems was to use the attributes of unicode [1] to define the attributes of each of the *graphemes*. These attributes are then used to map the character into a root grapheme and associated attributes [10]. For example, for Kazakh, which is a mixture of Cyrillic and Latin scripts, a subset of the graphemes associated with the letter "I" are

i G6;D2D3D6 LATIN SMALL LETTER I
I G6;D8D3D6 LATIN CAPITAL LETTER I
и G6;D1D2D3 CYRILLIC SMALL LETTER I
ѝ G6;D1D2D3D4 CYRILLIC SMALL LETTER I WITH GRAVE
й G6;D1D2D3D5 CYRILLIC SMALL LETTER SHORT I

where the following attributes are defined

D1 CYRILLIC D2 SMALL D3 LETTER D4 WITH GRAVE
D5 SHORT D6 LATIN D8 CAPITAL

All graphemes are thus mapped into a set of core graphemes, and attributes associated with the set of graphemes. This mimics the set of attributes associated with phones that can be obtained for all phones using, for example, X-SAMPA phonetic look-up tables.

The above scheme has assumed that all unicode characters have a distinct acoustic realisation. Unicode characters that do not have an acoustic realisation, or alter the realisation of an adjoining grapheme, can be split into two distinct groups. The first set are language-dependent graphemes, and are related to diacritics, but written as separate unicode characters, denoted by the word SIGN in the character descriptor. Note VOWEL SIGN characters in for example Abugida written languages are kept as separate symbols with acoustic realisations. In addition to the unicode attributes additional markers indicating the position of the grapheme in the word (beginning/middle/end) was added.

Language	Id	Script	TER (%)		
			Phon	Grph	CNC
Tok Pisin	207	Latin	40.6	41.1	39.4
Kazakh	302	Cyrillic/Latin	53.5	52.7	51.5
Telugu	303	Telugu	69.1	69.5	67.5

Table 1. *Babel FLP Tandem-SAT Performance: with confusion network (CN) decoding and* `CNC` *CN-combination.*

Table 1 contrasts the performance of three OP2 languages with scripts ranging from Latin (Tok Pisin) mixed Latin and Cyrillic (Kazakh) to Telugu (Telugu). The performance of the graphemic and the phonetic system is comparable

even though no phonetic information was used for the graphemic system. Additionally the combination of the phonetic and graphemic systems using Confusion Network Combination (CNC) [7] shows consistent gains.

2.2 Stimulated Training of Neural Networks

One of the issues with the standard training of neural networks is that the nodes are not interpretable. This lack of interpretability can cause issues for speaker adaptation and network generalisation as it is difficult to relate weights from the network to each other. To address this problem *stimulated network training* has been proposed [29,32,25]. The aim of stimulated training is to train networks where nodes with similar activation functions are grouped together. The Babel program, requiring low resources systems, should be suited to this form of training.

In stimulated training a phone (or grapheme) dependent prior distribution is defined over the normalised activation function outputs for each of the layers. The nodes in each layer are reorganised into a grid, so that each node, i, of a layer is represented as a point in a two dimensional *network-grid* space, $\boldsymbol{s}_i$. A point in this network-grid space is also associated with each phone $\boldsymbol{s}_p$. It is then possible to define a normalised distance from every node to the correct phone position. These normalised distances are used as a prior over the distribution of the activation function values for a layer. This prior encourages activation functions in the same locality to have the same normalised output.

To implement stimulated training, a regularisation term, $\mathcal{R}(\boldsymbol{\lambda})$, is added to the training criterion

$$\mathcal{F}(\boldsymbol{\lambda}) = \mathcal{L}(\boldsymbol{\lambda}) + \alpha\mathcal{R}(\boldsymbol{\lambda})$$

where $\mathcal{L}(\boldsymbol{\lambda})$ is the standard training criterion for parameters $\boldsymbol{\lambda}$, for an L hidden-layer network $\boldsymbol{\lambda} = \{\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(L)}\}$, α determines the contribution of the prior, $\mathcal{R}(\boldsymbol{\lambda})$. Here $\mathcal{R}(\boldsymbol{\lambda})$ is based on the KL-divergence of the prior distribution over the normalised activation, $g(\boldsymbol{s}_i, \hat{\boldsymbol{s}}_{p_t})$ and the current distribution, $\overline{h}_{ti}^{(l)}$. Thus

$$\mathcal{R}(\boldsymbol{\lambda}) = \sum_t \sum_l \sum_i g(\boldsymbol{s}_i, \hat{\boldsymbol{s}}_{p_t}) \log \left(\frac{g(\boldsymbol{s}_i, \hat{\boldsymbol{s}}_{p_t})}{\overline{h}_{ti}^{(l)}} \right)$$

where the two distributions are defined as:

1. *phone-specific activation distribution prior*: $g(\boldsymbol{s}_i, \hat{\boldsymbol{s}}_{p_t})$ is the normalised distance of a node and the current active-phone position. For these experiments:

$$g(\boldsymbol{s}_i, \hat{\boldsymbol{s}}_{p_t}) = \frac{\exp\left(-\frac{1}{2}||\boldsymbol{s}_i - \hat{\boldsymbol{s}}_{p_t}||^2\right)}{\sum_j \exp\left(-\frac{1}{2}||\boldsymbol{s}_j - \hat{\boldsymbol{s}}_{p_t}||^2\right)}$$

   where $\boldsymbol{s}_i$ the position in the network-grid space of node i, $\hat{\boldsymbol{s}}_{p_t}$ the position in the network-grid space of the "correct" phone at time t. The denominator summation is over all nodes in network layer l.

2. *network activation distribution*: $\overline{h}_{ti}^{(l)}$ is the normalised activation function output for node i of layer l at time instance t

$$\overline{h}_{ti}^{(l)} = \frac{\beta_i^{(l)} h_{ti}^{(l)}}{\sum_j \beta_j^{(l)} h_{tj}^{(l)}}; \qquad \beta_i^{(l)} = \sqrt{\sum_k w_{ik}^{(l+1)2}}$$

$h_{ti}^{(l)}$ is the output activation function value for node i of layer l at time instance t and $w_{ik}^{(l)}$ is the weight connection from node i of layer l to node k of layer $l+1$. $\beta_i^{(l)}$ is used to reflect the impact that the activation function has on the following layer, $l+1$ and has been found to be important for stimulated training.

This form of prior can be applied to any form of network. To generate the position of the correct grapheme, t-SNE was applied [32].

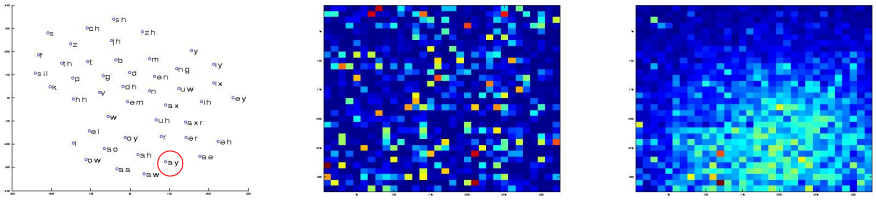

Fig. 2. Example of the impact of stimulated training for a particular instance of the phone `/ay/`. The left plot is the position of the stimulation points (`/ay/` circled), the center plot standard, unstimulated, training, the right plot stimulated training.

Figure 2 shows the impact of stimulated training using phone stimulation points on network training. The stimulation points, left figure, were obtained using t-SNE projections of phone feature means. The center plot shows the (scaled) network activation functions for hidden layer 3 (of 5) with standard training and has no structure in the node activation function values. This is expected for a randomly initialised distributed representation. The right plot shows the impact of stimulated training. Structure is clearly visible in the form of the activation functions [1].

Table 2 shows the impact of stimulated training on both ASR and KWS performance. For these results both Tandem and Hybrid systems were combined using joint decoding with stimulated training only being applied to the Hybrid systems. See [25] for additional system configurations and results. For all language investigated stimulated training gave performance gains for both ASR and KWS.

[1] For a complete movie of the activation functions for stimulated training see:

http://mi.eng.cam.ac.uk/∼mjfg/bneStimu.avi

Language Id	Stimu Train	TER (%)	MTWV iv	MTWV oov	MTWV tot
Amharic 307	✗	41.1	0.6500	0.5828	0.6402
	✓	40.8	0.6619	0.5935	0.6521
Javanese 402	✗	50.9	0.4991	0.4448	0.4924
	✓	50.7	0.5024	0.4679	0.4993

Table 2. Impact of stimulated training on ASR and KWS performance for the OP3 languages.

2.3 Web Data and RNN Language Models

One of the major issues associated with low-resource languages is the lack of appropriate language model training data. This has two immediate impacts. First if only 40-80 hours of transcriptions are available the resulting vocabulary will be very small resulting in high OOV rates for both ASR and KWS. Second the robustness of the estimates of the language model probabilities will be poor. To address this problem the web was "scraped" for data of the target language [20,34]. This allows large amounts of training data to be collected for many languages, with some exceptions such as Dholou. For example for Pashto the amount of data available from the FLP was 535K words, but 105M words could be collected from the web.

Unfortunately the availability of large amounts of data introduces two additional problems. Current state-of-the-art language models are built using Recurrent Neural Networks (RNNs) [21]. These models can take a significant amount of time to train on large amounts of data. To enable rapid deployment of systems it is necessary to improve the training time. Second the data collected from the web is typically poorly matched to the target domain, CTS. To address these problems modified training criteria were examined and "fine-tuning" to the FLP data used [3].

The standard training criterion for training neural network language models, including RNN LMs is based on cross-entropy. For word sequence $\boldsymbol{\omega}_{1:L} = \omega_1, \ldots, \omega_L$, the following criterion is optimised.

$$\mathcal{F}_{\mathsf{ce}} = -\frac{1}{L}\sum_{i=1}^{L} \log\left(P(\omega_i|\tilde{\boldsymbol{h}}_{i-1})\right)$$

Though this can be efficiently implemented using GPUs if the output layer, the prediction vocabulary size, is not large. As the size of the output vocabulary increases the computational cost is dominated the softmax normalisation term $Z(\tilde{\boldsymbol{h}}_{i-1})$

$$P(\omega_i|\tilde{\boldsymbol{h}}_{i-1}) = \frac{1}{Z(\tilde{\boldsymbol{h}}_{i-1})}\exp\left(\boldsymbol{w}_{f(\omega_i)}^{\mathsf{T}}\tilde{\boldsymbol{h}}_{i-1}\right)$$

This impacts both the training and decoding. For the word-based systems the vocabulary associated when including the web-data, was very large, for Pashto

273K words. Training the RNNLM using CE, the standard criterion, and then fine-tuning to the FLP data, is impractical for the surprise language evaluation as both training and recognition is very slow. To address this problem two different training criteria were investigated:

– *variance regularisation* (VR): this ensures that the normalisation term (in the prediction) is approximately constant for all word histories. An additional regularisation term is added to the standard cross-entropy (CE) criterion, $\mathcal{F}_{\texttt{ce}}$. The following criterion is optimised

$$\mathcal{F}_{\texttt{vr}} = \mathcal{F}_{\texttt{ce}} + \frac{\gamma}{2}\frac{1}{L}\sum_{i=1}^{L}\left(\log(Z(\tilde{\boldsymbol{h}}_{i-1})) - \overline{\log(Z)}\right)^2$$

If all the normalisation terms for all histories are constrained to be the same, it is therefore not necessary to compute it during recognition time, improving decoding time significantly;

– *noise contrastive estimation* (NCE): this trains a discriminative model between classifying the word sequences and noise samples often generated by a unigram language model. Here the following discriminative criterion is optimised

$$\mathcal{F}_{\texttt{nce}} = -\frac{1}{L}\sum_{i=1}^{L}\left(\log(P(y_i = \texttt{T}|\omega_i, \tilde{\boldsymbol{h}}_{i-1}) + \sum_{j=1}^{k}\log(P(y_i = \texttt{F}|\hat{\omega}_{ij}, \tilde{\boldsymbol{h}}_{i-1})\right)$$

where $\hat{\omega}_{ij}$ are *noise samples* for ω_i, often generated by a uni-gram language-model. In this model it is not necessary to estimate the normalisation term during training or recognition.

Language	Id	Vocab	RNN Crit		Time (hrs)		TER	MTWV		
			Trn	F-T	Train	Rescore	(%)	iv	oov	tot
Pashto	104	14.4K	—		—		44.1	0.4808	0.2412	0.4541
		376.3K	—		—		43.8	0.4828	0.4083	0.4750
			CE	CE	125.0	23.0	42.8	0.4975	0.4048	0.4871
			NCE	VR	10.7	2.0	43.0	0.4975	0.3953	0.4862

Table 3. Impact of web RNN LMs on KWS performance on Pashto. The RNN-Criterion is either used for initial training (Trn) or fine-tuning (F-T).

Table 3 shows the impact of web-data on the ASR and KWS systems. For details of the system configurations see [3]. The first line is the performance when only using the FLP data to train an tri-gram language model. Comparing the first and second lines, where a language model component trained on the web-data was used, shows that the use of the web-data had little impact on ASR performance. However the KWS performance is significantly better. The main

reason for this is the performance on OOV KWS terms (as defined by the FLP), as the increased vocabulary reduces the need to use "phone" search for the KWS system.

Table 3 also shows the impact of including an RNNLM in the system. For both configurations the RNNLM was initially trained on all the data and then fine-tuned to the FLP data. Line three of the table shows the performance when CE is used for both stages. Gains can be see for both ASR and KWS performance. However examining both the training and lattice rescoring times shows that it takes over 5 days to train the system. Using NCE for initial training, and then VR to fine-tune, reduced this training time by more than an order-of-magnitude, and similarly for the deciding time.

3 Improved ASR and KWS Efficiency Research at CUED

As previously discussed to minimise the impact of error propagation from the ASR system to the KWS very large lattices are created. Additionally to maximise performance multiple systems need to be combined together to yield the final result. The combination of the two can result in significant computational cost when handling large quantities of data. This section briefly describes two approaches that were used at CUED to reduce the computational cost: unique arc-per-second pruning to reduce the size of the lattices; and model-merging to reduce the number of ASR and KWS runs.

3.1 Unique-Arcs Per-Second Pruning

There are two contradictory requirements for the lattices that are used for KWS. First they should be large and diverse, containing multiple competing paths. Second, they should be compact so that the speed of KWS is fast. There have been a number of approaches adopted to balance these two, including confusion network based KWS and ensuring that all words in the best path for all word sequences are kept. An alternative approach that is to control the distribution of the number of unique arcs at each time instance. It is possible to apply this process during decoding, or on lattices. The basic process is:

For each selected time instance:

- for each word (unique arc) rank order all arcs by score for that word;
- rank order all words by the best arc score for that word;
- prune arcs so that the selected distribution over words (unique arcs) is satisfied, ensuring that connections to all arcs in the previous pruning time instance are maintained.

This approach is highly flexible as it is possible to control the size of lattice by varying the target unique-arcs-per-second distribution. Figure 3 shows the impact of UAPS pruning. The top figure shows the total number of arcs in a

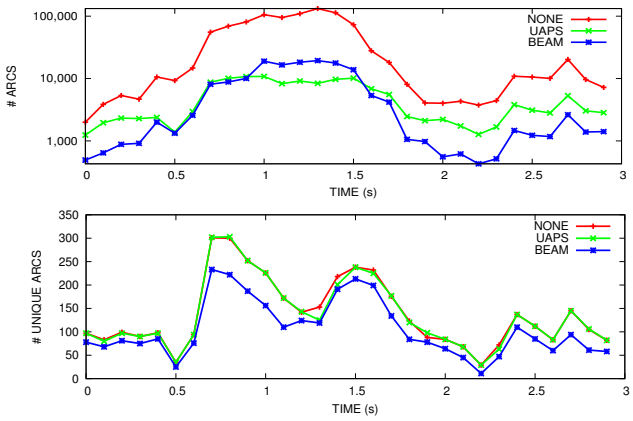

Fig. 3. Illustration of unique-arcs-per-second pruning for a sentence

lattice. The lower figure shows the number of unique arcs. The standard beam-pruning and UAPS pruning are configured to yield the same lattice size for the utterance. It is clear from the diagram that UAPS pruning maintains the number of unique arcs of the original unpruned system, but at a significantly smaller lattice size.

All the CUED evaluation systems for OP2 and OP3 were based on UAPS pruning. Typically the size of lattices was reduced by an order of magnitude, with no impact on KWS performance.

Language	Id	Arcs/Sec	
		Decode	UAPS
Mongolian	401	88,479	17,623
Javanese	402	41,880	11,109

Table 4. Example lattice size (arcs/second) of the original lattices and after unique-arcs per-second pruning

Table 4 shows the sizes of lattice generated from the decoding process and after UAPS for two of the OP3 languages. For these systems the number of unique arcs is approximately the same. The size of lattices after UAPS are approximately an order of magnitude smaller, dramatically improving time/memory requirements for KWS.

3.2 Model Merging or Posting-List Merging

As previously discussed one important approach to improve the performance of both ASR and KWS is to perform system combination. Here multiple, preferably complementary systems, are combined together to yield the final result.

An important consideration for these approaches is the computational load. The simplest approach is to run all the systems separately and combine the final outputs together. This is the approach adopted with ROVER [8] and CNC [7] for ASR, and posting-list merging for KWS [19]. If four systems were to be combined this would requiter four ASR decodes, followed by four KWS runs.

To reduce the computational load it is possible to combine the multiple systems together. The approach adopted at CUED for the Babel programme was log-linear model-combination [2,33]. Here the log-likelihood of a particular observation $\mathbf{o}_t$ for state $\mathtt{s}$ is given by

$$\log(p(\mathbf{o}_t|\mathtt{s})) = \frac{1}{Z(\mathbf{o}_t)} \exp\left(\sum_{m=1}^{M} \alpha_m \log(p(\mathbf{o}_t|\mathtt{s};\boldsymbol{\lambda}^{(m)}))\right)$$

where $\log(p(\mathbf{o}_t|\mathtt{s};\boldsymbol{\lambda}^{(m)}))$ is the log-likelihood from model m and α_m is the related weight. Only a single decode and lattice generation, and KWS search are performed. As the normalisation term, $Z(\mathbf{o}_t)$, is only a function of the observation it does not impact the rank ordering in decoding. Thus the weight for each model α_m can be hand selected and used for decoding. This was the approach adopted here [31].

An interesting question is the nature of the systems to combine. For OP2 evaluation systems Tandem and Hybrid systems were combined. For OP3 multiple multi-lingual bottleneck features (BN) were made available from Aachen (A28) and IBM (I28), see [4] for details. Two configurations were compared. First the use of joint decoding between Tandem and Hybrid systems for each of the BN features (labelled A28 and I28) and then output combination (CNC for ASR and posting-list merging for KWS), or joint decoding using all fours models was investigated. The weights for the models were empirically selected, but consistent for all languages.

Language	Id	System comb.	TER (%)	MTWV		
				iv	oov	tot
Javanese	402	A28	52.5	0.4787	0.4379	0.4736
		I28	52.1	0.4763	0.4283	0.4712
		A28⊕I28	50.9	0.4991	0.4448	0.4924
		A28⊗I28	50.9	0.4979	0.4843	0.4970

Table 5. Performance of the OP2 joint decoding configuration using the Aachen BN features (A28) and the IBM BN features (I28), ⊕ indicates CNC/posting-list merging, ⊗ indicated joint decoding.

The performance of various configurations is shown in Table 5. Though performing two separate runs and then two KW searches yielded better performance (for KWS) for all languages the differences were not large compared to the memory and computational loads. For the final evaluation joint decoding over all four systems was used for efficiency.

4 Language Analysis and Prediction

There is a wide range of performance over the languages distributed in the BP, OP1, OP2 and OP3. One of the interesting aspects of these language packs is that they are consistently annotated and are associated with challenging data, CTS. An interesting question is whether it is possible to predict the performance for a particular language without having to build a full system.

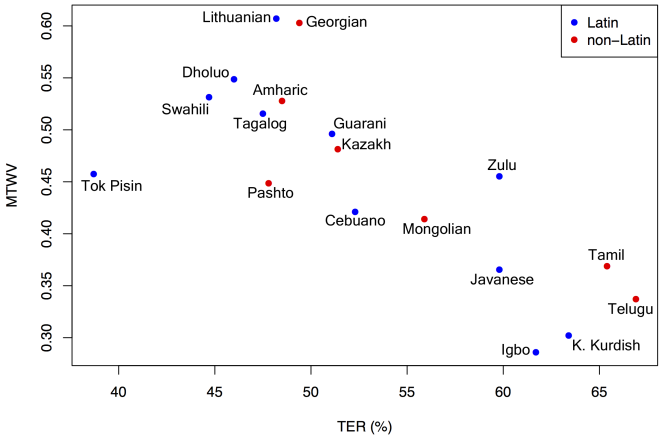

Fig. 4. *Summary plot (MTWV vs TER) for FLP systems in a single graphemic system OP2 configuration.*

As a starting point for the analysis a sub-set of languages were all built using a consistent, relatively advanced graphemic lexicon configuration that was used for the OP2 FLP evaluation [31]. The subset was selected to give a spread over the language groups and interesting language pairs for analysis. For example: the Dravidian Languages, Tamil and Telugu generally performed poorly; members of the Niger-Kongo languages, Swahili, Zulu and Igbo, were selected as there was a large performance difference (ASR/KWS) between Swahili and Zulu, and the opportunity to predict Igbo. Figure 4 shows the plot of MTWV against TER for all these languages. It can be seen that MTWV and TER are negatively correlated, with some outliers: Tok Pisin has a worse than expected KWS performance given the TER; Zulu, Lithuanian and Georgian have a better KWS performance than expected. What is also interesting is the wide spread of performance ranging from 65% to 40% TER for ASR, and 0.30 to 0.60 MTWV for KWS.

A range of attributes (including SNR, number of words/phones, OOV, LM perplexity) from the languages were evaluated to see whether performance could be predicted using language attributes rather than building systems. Unfortunately none of these showed strong correlations with the final performance. One issue that impacts this form of analysis is the level of inter-annotator disagreement (WER), which can vary considerably over languages: Tamil $\approx 25\%$; Lithuanian $\approx 10\%$.

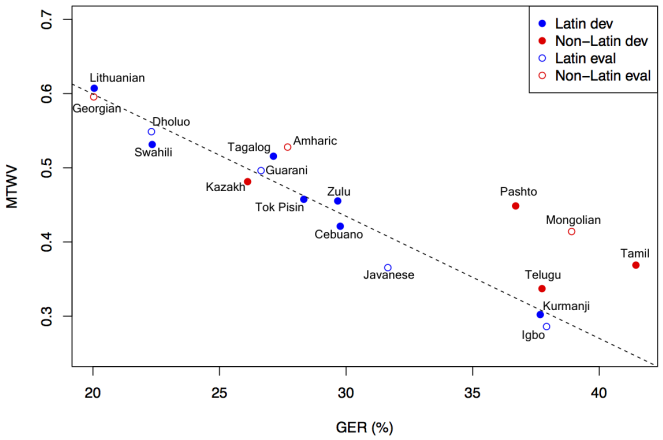

Fig. 5. Summary plot (MTWV vs Graphemic Error Rate (GER)) for FLP systems in an OP2 graphemic system configuration. Blue indicates Latin script alphabet. Best fit line computed using the 7 dev Latin script languages.

Rather than looking at general language attributes, the ability to predict final ASR and KWS performance from a simple initial ASR build was investigated. The most informative attribute was Graphemic Error Rate (GER) [2] calculated from a simple maximum likelihood PLP-based, speaker-independent, graphemic system on the training data. The system is relatively fast to train, and no held-out data is required. Additionally inter-annotator inconsistencies are implicitly handled. Figure 5 shows MTWV against GER. For all the Latin script languages there is a strong correlation between GER and MTWV. The previous outlier Latin script languages, Tok Pisin, Zulu and Lithuanian, are now in-line with predictions. As the grapheme accuracy will depend on the number of graphemes the non-Latin script languages marked in red (Kazakh, Tamil, Telugu

[2] All markers such as accents are stripped from the grapheme to yield the root grapheme. Thus Latin scripts have 26 graphemes. These accuracies include silence at the beginning and end of sentences, and between all words.

and Pashto) are not considered. These are expected to have a higher grapheme error rate as there are more graphemes present (Kazakh is an exception).

Language	Id	Script	%TER		MTWV	
			pred	obs	pred	obs
Dholuo	403	Latin	45.4	46.0	0.561	0.549
Guarani	305		49.5	51.1	0.490	0.496
Igbo	306		60.2	61.7	0.304	0.286
Javanese	402		54.2	59.8	0.408	0.362
Amharic	307	Ethiopic	50.5	48.5	0.473	0.528
Mongolian	401	Cyrillic	61.1	55.9	0.288	0.414
Georgian	404	Mkhedruli	43.3	49.2	0.599	0.596

Table 6. Predictions of OP2 configuration ASR and KWS performance for OP3 languages, including OP3 evaluation language Georgian (404).

It is also interesting to examine the ability to predict performance using held-out data. the OP3 languages were treated as the held-out data, and all the remaining Latin script languages used to generator a predictor. For MTWV this is the dotted line in Figure 5. Similarly a linear predictor for TER was generated. Table 6 shows the resulting predictions, and actual observed values. For the Latin script it can be seen that GER is a good predictor for MTWV and indicative for TER.

5 Conclusions

This paper has briefly outlined some of the approaches developed at Cambridge University to handle low-resource keyword-spotting and speech recognition under the Babel programme. Two distinct areas are discussed low-resource speech recognition and efficient low-resource keyword-spotting. The final section of the paper briefly examines the performance on a wide-range of languages in a consistent configuration and how to predict performance.

Though significant advances were made during the Babel programme on low-resource speech processing, it still remains a highly challenging area. The current ability to leverage data across languages is limited despite the fact that there is a common acoustic generation process for all languages, human physiology. Additionally many languages have common syntactic and semantic structure. Extracting these, and leveraging connections is still not addressed.

Acknowledgements

This work was supported in part by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Research Laboratory (DoD/ARL)

contract number W911NF-12-C-0012. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government. This work made use of data provided by IARPA [3].

The authors would like to thank the contributions of all the members of the CUED Babel team during the project. In particular Dr X. Chen, J. Vasilakes, Dr H. Wang and Dr S. Rath who directly worked on the evaluation systems and during the "interesting" Babel evaluation periods. The authors would also like to thank all the members of the LORELEI team, in particular the IBM and RWTH Aachen Babel teams.

References

1. The unicode consortium. httlp://http://www.unicode.org, accessed: 2014-09-30
2. Beyerlein, P.: Discriminative model combination. In: Proc. ASRU (1997)
3. Chen, X., Ragni, A., Liu, X., Vasilakes, J., Knill, K.M., Gales, M.J.: Recurrent neural network language models for keyword search. In: ICASSP (2017)
4. Cui, J., Kingsbury, B., Ramabhadran, B., Sethy, A., Audhkhasi, K., Cui, X., Kislal, E., Mangu, L., Nussbaum-Thom, M., Picheny, M., et al.: Multilingual representations for low resource speech recognition and keyword search. In: Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on. pp. 259–266. IEEE (2015)
5. Cui, X., Goel, V., Kingsbury, B.: Data augmentation for deep neural network acoustic modeling. IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP) 23(9), 1469–1477 (2015)
6. Evermann, G., Woodland, P.C.: Large vocabulary decoding and confidence estimation using word posterior probabilities. In: Proc. of ICASSP (2000)
7. Evermann, G., Woodland, P.: Posterior probability decoding, confidence estimation and system combination. In: Proc. Speech Transcription Workshop. vol. 27, p. 78. Baltimore (2000)
8. Fiscus, J.G.: A post-processing system to yield reduced word error rates: Recogniser Output Voting Error Reduction (ROVER). In: Proc ASRU (1997)
9. Fiscus, J.G., et al.: Results of the 2006 Spoken Term Detection Evaluation. In: Proc. ACM SIGIR Workshop on Searching Spontaneous Conversational Speech (2007)
10. Gales, M.J., Knill, K.M., Ragni, A.: Unicode-based graphemic systems for limited resource languages. In: ICASSP. pp. 5186–5190. IEEE (2015)
11. Gales, M.J., Knill, K.M., Ragni, A., Rath, S.P.: Speech recognition and keyword spotting for low-resource languages: Babel project research at cued. In: SLTU. pp. 16–23 (2014)
12. Grezl, F., Karafiat, M., Janda, M.: Study of probabilistic and bottle-neck features in multilingual environment. In: Proc. ASRU (2011)
13. Harper, M.: IARPA Babel Program, http://www.iarpa.gov/Programs/ia/Babel/babel.html

[3] The following data was used in the FLP configuration: IARPA-babel106-v0.2f, IARPA-babel202b-v1.0d, IARPA-babel204b-v1.1b, IARPA-babel205b-v1.0a, IARPA-babel206b-v0.1d, IARPA-babel207b-v1.0a, IARPA-babel301b-v1.0b, IARPA-babel302b-v1.0a, IARPA-babel303b-v1.0a, IARPA-babel304b-v1.0b, IARPA-babel104b-v0.4bY, IARPA-babel306b-v2.0c, IARPA-babel401b-v2.0b, IARPA-babel402b-v1.0b, IARPA-babel403b-v1.0b, IARPA-babel404b-v1.0a.

14. Hartmann, W., Ng, T., Hsiao, R., Tsakalidis, S., Schwartz, R.: Two-stage data augmentation for low-resourced speech recognition. Interspeech 2016 pp. 2378–2382 (2016)
15. Hermansky, H., Ellis, D., Sharma, S.: Tandem Connectionist Feature Extraction for Conventional HMM Systems. In: Proc. ICASSP (2000)
16. Hinton, G., Deng, L., et al.: Deep Neural Networks for Acoustic Modeling in Speech Recognition. Signal Processing Magazine, IEEE 29(6), 82–97 (Nov 2012)
17. Kanthak, S., Ney, H.: Context-dependent acoustic modelling using graphemes for large-vocabulary speech recognition. In: Proc. ICASSP (2002)
18. Killer, M., Stüker, S., Schultz, T.: Grapheme based speech recognition. In: Proc. EUROSPEECH (2003)
19. Mamou, J., Cui, J., Cui, X., Gales, M.J., Kingsbury, B., Knill, K., Mangu, L., Nolden, D., Picheny, M., Ramabhadran, B., et al.: System combination and score normalization for spoken term detection. In: ICASSP. pp. 8272–8276. IEEE (2013)
20. Mendels, G., Cooper, E., Soto, V., Hirschberg, J., Gales, M.J., Knill, K.M., Ragni, A., Wang, H.: Improving speech recognition and keyword search for low resource languages using web data. In: INTERSPEECH. pp. 829–833 (2015)
21. Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., Khudanpur, S.: Recurrent neural network based language model. In: Interspeech. vol. 2, p. 3 (2010)
22. Miller, D.R.H., Kleber, M., et al.: Rapid and accurate spoken term detection. In: Proc. Interspeech (2007)
23. Ragni, A., Knill, K.M., Rath, S.P., Gales, M.J.F.: Data augmentation for low resource languages. In: Proc InterSpeech (2014)
24. Ragni, A., Dakin, E., Chen, X., Gales, M.J., Knill, K.M.: Multi-language neural network language models. In: Interspeech. vol. 8, pp. 3042–3046 (2016)
25. Ragni, A., Wu, C., Gales, M.J., Vasilakes, J., Knill, K.M.: Stimulated training for automatic speech recognition and keyword search in limited resource conditions. In: ICASSP (2017)
26. Rath, S.P., Knill, K.M., Ragni, A., Gales, M.J.: Combining tandem and hybrid systems for improved speech recognition and keyword spotting on low resource languages. In: INTERSPEECH. pp. 835–839 (2014)
27. Swietojanski, P., Ghoshal, A., Renals, S.: Revisiting hybrid and gmm-hmm system combination techniques. In: ICASSP. pp. 6744–6748. IEEE (2013)
28. Szoke, I., Burget, L., Cernocky, J., Fapso, M.: Sub-word modeling of out of vocabulary words in spoken term detection. In: Proc. of SLT (2008)
29. Tan, S., Sim, K.C., Gales, M.: Improving the interpretability of deep neural networks with stimulated learning. In: Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on. pp. 617–623. IEEE (2015)
30. Vergyri, D., Shafran, I., et al.: The SRI/OGI 2006 spoken term detection system. In: Proc. Interspeech (2007)
31. Wang, H., Ragni, A., Gales, M.J., Knill, K.M., Woodland, P.C., Zhang, C.: Joint decoding of tandem and hybrid systems for improved keyword spotting on low resource languages. In: INTERSPEECH. pp. 3660–3664 (2015)
32. Wu, C., Karanasou, P., Gales, M.J., Sim, K.C.: Stimulated deep neural network for speech recognition. In: Proc. Interspeech. pp. 400–404 (2016)
33. Yang, J., Zhang, C., Ragni, A., Gales, M.J., Woodland, P.C.: System combination with log-linear models. In: ICASSP. pp. 5675–5679. IEEE (2016)
34. Zhang, L., Karakos, D., Hartmann, W., Hsiao, R., Schwartz, R., Tsakalidis, S.: Enhancing low resource keyword spotting with automatically retrieved web documents. In: Interspeech (2015)