Optimised Code-Switched Language Model Data Augmentation in Four Under-Resourced South African Languages

Joshua Jansen van Vueren(✉) and Thomas Niesler

Department of Electrical and Electronic Engineering, Stellenbosch University, Stellenbosch, South Africa
{jjvanvueren,trn}@sun.ac.za

Abstract. Code-switching in South African languages is common but data for language modelling remains extremely scarce. We present techniques that allow recurrent neural networks (LSTMs) to be better applied as generative models to the task of producing artificial code-switched text that can be used to augment the small training sets. We propose the application of prompting to favour the generation of sentences with intra-sentential language switches, and introduce an extensive LSTM hyperparameter search that specifically optimises the utility of the artificially generated code-switched text. We use these strategies to generate artificial code-switched text for four under-resourced South African languages and evaluate the utility of this additional data for language modelling. We find that the optimised models are able to generate text that leads to consistent perplexity and word error rate improvements for all four language pairs, especially at language switches. This is an improvement on previous work using the same speech data in which text generated without such optimisation did not provide improved performance. We conclude that prompting and targeted hyperparameter optimisation are an effective means of improving language model data augmentation for code-switched speech recognition.

Keywords: Code-switching · Language model data augmentation · LSTM · Speech recognition · Under-resourced languages · African languages · Bantu languages

1 Introduction

Code-switching is the use of more than one language within and between sentences, a phenomenon that is pervasive in multilingual countries such as South Africa. Language switches are known to occur on the intra-word (prefix/suffix), word (insertional), and phrase (alternational) level [22]. One can further distinguish between intra-sentential, and inter-sentential switching [7]. Accurate modelling of code-switching is challenging due to its spontaneous nature and the severe lack of data.

© Springer Nature Switzerland AG 2021
A. Karpov and R. Potapova (Eds.): SPECOM 2021, LNAI 12997, pp. 1–14, 2021.
https://doi.org/10.1007/978-3-030-87802-3_28

Various strategies for the augmentation of code-switched data have been explored and can be categorised as neurally generative and syntactically constrained. Generative adversarial networks (GANs), part-of-speech (POS) tags [5], and hybrid BERT-GAN architectures [9] have been used to generate code-switched sentences from a monolingual Mandarin corpus either by learning which words or phrases should be translated into English or by masking tokens one-by-one in the input sequence. The synthesized datasets were pooled with the code-switched training sets and used to train RNN and n-gram language models. Although it was found that the n-gram perplexity worsened, when used in ASR experiments the performance was slightly improved. The RNN language model was evaluated solely using perplexity and improved over a baseline result.

In [16], a code-switched corpus is synthesized by aligning and embedding words and phrases from parallel monolingual English and Mandarin text according to the matrix language frame (MLF) theory [17]. A code-switch probability is assigned to each pair of aligned phrases and used to generate code-switched sequences for LSTM pre-training. This pre-training improved perplexity, lead to faster convergence, and improved ASR performance in lattice rescoring. Aligned parallel English-Mandarin text was also used in [13] to generate code-switched sentences that were subsequently fused with data generated by a pointer-generator neural network. A RNN composed of two LSTMs that separately model monolingual segments was shown to reduce perplexity in [10], where the current language is used as a signal to select the LSTM which models the next word. In [20] speech synthesized using novel algorithms and code-switched text generated using equivalence constraint theory (ECT) were incorporated into the training data, thereby improving English-Hindi ASR performance. With a similar aim, [24] improved perplexities and word error rates for Dutch-Frisian code-switching through the augmentation of acoustic and text data. Large artificial code-switched corpora were generated using an LSTM trained on a small amount of in-domain code-switched text. Text was also manually translated from Dutch to Frisian utilising an open-source web-service. Finally, ASR transcriptions were incorporated into the training data. In [23] code-switched bigrams synthesised using word embeddings were shown to improve both perplexity and speech recognition performance at language switches. In further work, an LSTM was used to augment English-isiZulu data, thereby decreasing perplexity [3].

Here, we build on the work described in [2,3,24]. We also train LSTMs on severely under-resourced code-switched data and then use these to generate artificial text with which to augment the training set. However, we extend these approaches by introducing the targeted optimisation of the text generation to produce sequences with useful code switches. To do this, we apply an extensive hyperparameter search that optimises text generation quality. Such targeted hyperparameter optimisation for code-switches has not been reported on before.

The remainder of this paper is organised as follows. Section 2 describes the datasets utilised for experimentation. Section 3 details the experimental setup and Sect. 4 presents the results. Finally, Sect. 5 concludes.

2 Dataset

This work uses a manually transcribed corpus of code-switched speech compiled from South Africa soap opera episodes [21]. The speech in this corpus includes four South African Bantu languages (isiZulu, isiXhosa, Sesotho and Setswana) as well as English. Table 1 shows the subdivision of the corpus into four bilingual sub-corpora: English-isiZulu (EZ), English-isiXhosa (EX), English-Sesotho (ES), and English-Setswana (ET).

Table 1. The soap opera corpus, showing the total number of word tokens, word types, and code switches in the four bilingual sub-corpora. CS_{EB} indicates the number of switches from English to a Bantu language, while CS_{BE} indicates switches from Bantu to English. The final column indicates the duration of the audio data.

Pair	Partition	English		Bantu		Total				
		Tok	Typ	Tok	Typ	Tok	Typ	CS_{EB}	CS_{BE}	Dur
EZ	Train	28033	3608	24350	6788	52383	10396	2236	2743	4.81 h
	Dev	832	414	734	452	1566	866	175	198	8.00 min
	Test	2457	870	3199	1435	5656	2305	688	776	30.4 min
EX	Train	20324	2630	12215	5086	32539	7716	776	1003	2.68 h
	Dev	1153	484	1147	762	2300	1246	91	113	13.7 min
	Test	1149	498	1502	889	2651	1387	328	363	14.3 min
ES	Train	15395	2255	19825	2086	35197	4339	1565	1719	2.36 h
	Dev	843	437	2227	614	3067	1050	156	166	12.8 min
	Test	1794	659	2265	535	4054	1193	403	396	15.5 min
ET	Train	16180	2361	19570	1448	35725	3808	1885	1951	2.33 h
	Dev	1170	514	2539	539	3707	1052	224	251	13.8 min
	Test	1970	729	2979	526	4939	1254	505	526	17.8 min

All four Bantu languages are agglutinative. Furthermore, isiZulu and isiXhosa have a conjunctive orthography, leading to larger vocabularies (Table 1) than Sesotho and Setswana, which have a more disjunctive orthography.

Table 2. Out-of-domain monolingual corpora.

Language	English	isiZulu	isiXhosa	Sesotho	Setswana
Tokens	471M	3.25M	0.99M	0.23M	2.84M

We also use the five out-of-domain monolingual corpora outlined in Table 2. This data was gathered from transcriptions of conversations, newspaper reports and web text [2]. The table shows that, while substantial out-of-domain text resources are available for English, much less is available than the four under-resourced Bantu languages.

3 Experimental Strategy

Our generative LSTM [11,12] consists of 3 layers: an embedding layer, an LSTM layer and a dense layer. Gated recurrent units (GRUs) [6] and dropout [14] were found to be ineffective in preliminary experiments. Adam [15] was used for gradient descent and cross entropy as the optimisation criterion. All LSTMs were implemented using Tensorflow [1], all n-gram language models are trigrams with Witten-Bell discounting trained using the SRILM toolkit [19].

Speech recognition experiments were performed using KALDI [18] according to the procedure in [4]. The training procedure is broken into two sections: multilingual pre-training, and language adaptation. The multilingual pre-training trains a CNN-TDNN-F acoustic model on the pooled data from all four subcorpora as detailed in Table 1. The adaptation phase fine-tunes the pre-trained CNN-TDNN-F by training for two epochs on the relevant bilingual sub-corpus.

3.1 Metrics

In addition to perplexity (PP), to pinpoint the difficulties encountered at language switches, we also consider the code-switched perplexity (CPP) as first defined in [23]. CPP is the perplexity when calculated only across the language switches, while monolingual perplexity (MPP) is the perplexity calculated within monolingual stretches of text.

Similarly, in speech recognition experiments we make use of the code-switched bigram (CSBG) error, which is the speech recognition error computed only for words immediately following a language switch [3]. By observing this figure, the impact of our interventions specifically on code-switching can be assessed.

3.2 Text Synthesis and N-Gram Augmentation

To generate text, the start sequence token <s> is presented to the generative model as input. The model then returns a categorical probability distribution over the possible next words as well as the hidden and cell state vectors. Using random sampling, a next word is chosen from the categorical distribution. This word token and the hidden and cell state vectors are then presented to the model as a new input. This continues until the end sequence token </s> is synthesized, or the sequence length reaches a predetermined maximum limit.

$$p_i = \frac{\exp(z_i/\tau)}{\sum_j \exp(z_j/\tau)} \tag{1}$$

Random sampling allows more diverse samples to be obtained from the inner-representation of the LSTM. The unnormalized likelihoods (z_i) output by the dense layer can be focused or spread by multiplication with a heuristic temperature value τ, as shown in Eq. 1, where p_i is the resultant probability.

Prompting. Neural networks trained to accomplish downstream natural language processing (NLP) tasks are typically either pre-trained and adapted to a task-domain or 'prompted' by syntactic characters [8]. We apply prompting to our LSTM model in an attempt to generate code-switched sequences more effectively and reliably. During training, we mark all sequences containing code-switches with the special start sequence token $<s_{cs}>$, while monolingual sequences begin with token $<s_{mono}>$. In combination with this technique, the discarding of monolingual sequences (ablation) is also considered. This results in four text generation strategies, listed in Table 3.

Table 3. The four considered text generation strategies.

Strategy	Discard monolingual sequences (ablation)	Use special start token (prompting)
All_{Text}		
CS_{Text}	×	
All_{Prompt}		×
CS_{Prompt}	×	×

Evaluating the Quality of Artificially-Generated Text. To determine whether the synthesized data is useful, it is used to train an n-gram language model, which is linearly interpolated with a baseline n-gram language model (LM_B) trained only on the training set (Table 1). The interpolation weight λ is optimised to minimise the development set perplexity. During preliminary experiments it became apparent that the optimal perplexity of this interpolated model was generally not observed for text generated by LSTMs that had been trained until convergence on the development set, which was observed to be reached after only a few training epochs ($N_{\text{E}} \lesssim 10$). Instead, text generated using LSTMs for which training was allowed to continue after convergence provided greater perplexity improvements. Therefore, the number of training epochs was also considered to be a hyperparameter that was explicitly optimised.

3.3 Hyperparameter Tuning

Fig. 1 shows how optimization is split into three phases.

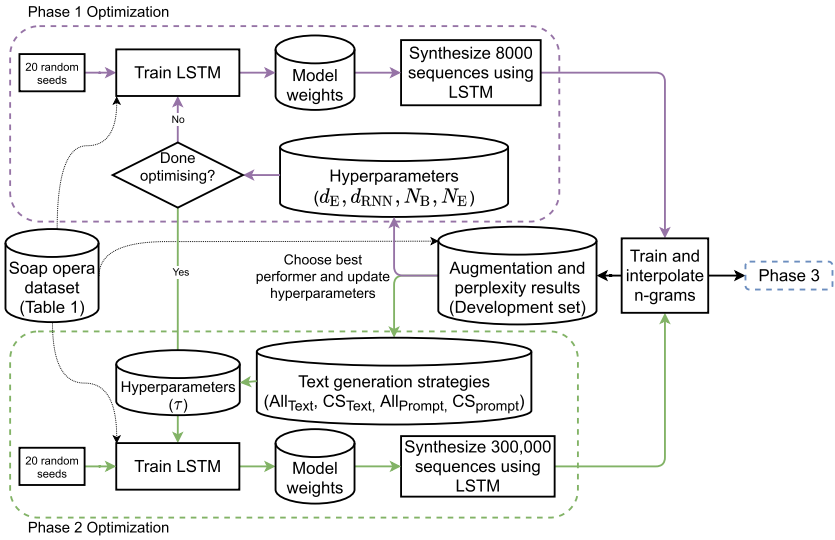

Fig. 1. Block diagram of the hyperparameter and text generation tuning strategy used to optimise the generative LSTMs.

Table 4. Hyperparameters considered for optimisation.

Symbol	Description	Considered range
N_B	Batch size	$\{8, 16, 32, 64, 128, 256\}$
d_{RNN}	NN dimension	$\{16, 32, 64, 128, 256, 512, 768\}$
d_E	Embedding dimension	$\{2, 4, 8, 16, 32, 64, 128, 256\}$
N_E	Training epochs	$\{5, 10, 15, ..., 60\}$
τ	Temperature	$\{0.75, 1.0, 1.25, 1.5\}$

Phase One: N_B, d_{RNN}, d_E and N_E. In Phase One, 20 separate LSTMs are initialised, each with a different random seed. These models are trained on the training sets in Table 1 and then used to generate 20 respective sets of artificial text, each 8000 sequences in length and without ablation (All$_{Text}$, Table 3). Each of these sets of text is used to train an n-gram language model that is subsequently interpolated with the baseline n-gram (LM$_B$) after which the perplexities described in Sect. 3.2 are calculated and stored. The hyperparameters in Table 4 are tuned one at a time, repeating the above procedure for each considered value while holding the other parameters constant. Optimal hyperparameter values were chosen based on the mean over the 20 experiments. Since the hyperparameters N_B, d_{RNN} and d_E influence the rate of convergence, the models are first evaluated for $N_E = \{20, 40, 60\}$ epochs. Subsequently, N_E is optimized over the range in Table 4. Due to the computational complexity of this process, only one pass for each hyperparameter was performed.

Phase Two: τ. Phase Two of the optimization process utilised the best performing hyperparameters from Phase One to train 20 new LSTM models for each of the four text generation strategies in Table 3. For each strategy, 300,000 sequences are synthesized and this synthesis is repeated for the temperatures τ listed in Table 4. The set of 20 models associated with the text generation strategy with the best average performance was then used to generate 20 final datasets consisting of 1 million sequences each. The 20 sets of 1 million sequences are used to train 20 n-gram language models, which are each interpolated with the baseline n-gram (LM_B). The interpolated n-gram with the lowest code-switched perplexity (CPP) on the development set is then selected to be used in the third and final phase of augmentation experiments.

Table 5. Phase Three language models. LM_B: Baseline, LM_{B+S}: Baseline interpolated with language model trained on synthesized text, LM_{B+M}: Baseline interpolated with language models trained on monolingual text, LM_{B+S+M}: Baseline interpolated with both synthesized and monolingual language models.

Label	Train	Synth	English	Bantu
LM_B	×			
LM_{B+S}	×	×		
LM_{B+M}	×		×	×
LM_{B+S+M}	×	×	×	×

Phase Three: Interpolation. In Phase Three, we consider the four augmented n-gram language model configurations listed in Table 5. The baseline (LM_B) is trained only on the training set (Table 1). LM_{B+S} indicates an interpolation between the n-gram trained on the training set (LM_B) and the best performing n-gram trained on the synthesized dataset of 1 million sequences from Phase Two. LM_{B+M} indicates an interpolation between LM_B and two n-gram language models each trained on the respective out-of-domain monolingual corporus shown in Table 2. LM_{B+S+M} indicates an interpolation between LM_B, an n-gram language model trained on the synthesized dataset, as well as the two n-gram language models trained on the out-of-domain monolingual corpora. The resulting four interpolated n-gram language models in Table 5 are used in ASR experiments.

4 Experimental Results

We note at the outset that, in our previous work on the same speech datasets, synthesizing text using an LSTM without the hyperparameter optimisation we proposed here afforded only insubstantial (<1%) improvements in perplexity over the baseline, and no improvements in speech recognition accuracy [2].

4.1 Hyperparameter Tuning

Table 6 shows the hyperparameter optimization process in Phase One (Sect. 3.3) for the English-isiZulu sub-corpus. Similar behaviour was observed for the other three language pairs. Each row reports the successive optimisation of the first four hyperparameters in Table 4. We see that interpolation with a language model trained on the synthesized dataset after optimising the first four hyperparameters leads to a 1.4% relative reduction in perplexity over the baseline LM_B. We expect only small perplexity improvements in Phase One due to the small amount of text generated. More substantial improvements are achieved in Phase 2 of the optimisation procedure. We also note from the table that code-switched perplexities (CPP) are much higher than overall perplexities (PP). This indicates the high degree of uncertainty the n-gram has as language switches. Reducing this uncertainty is the primary objective of our data augmentation.

Table 6. Development set perplexity (PP) and optimal average interpolation weight (λ) between the baseline LM_B and an n-gram trained on the synthesized text for the English-isiZulu sub-corpus. Each row indicates the optimization a successive hyperparameter (Par) and its best performing value (Value).

Par	Value	λ	Loss_{Train}	Loss_{Dev}	PP	CPP
LM_B	–	–	–	–	626	3226
N_B	32	0.94	2.46	4.32	622 ± 2.2	3257 ± 28.9
d_{RNN}	512	0.94	1.5	6.16	622 ± 1.6	3270 ± 27.8
d_{E}	64	0.9	1.25	5.87	616 ± 3.7	3235 ± 48.1
N_E	35	0.9	1.42	5.51	617 ± 2.5	3244 ± 36.4

We illustrate the effectiveness of the hyperparameter tuning strategy and the need to choose effective parameters in Fig. 2. The figure shows the average relative improvement over the four language pairs afforded by the language model incorporating the synthesized data relative to the baseline language model (LM_B) when optimising each hyperparameter. It is clear from the figure that each hyperparameter has an associated optimum.

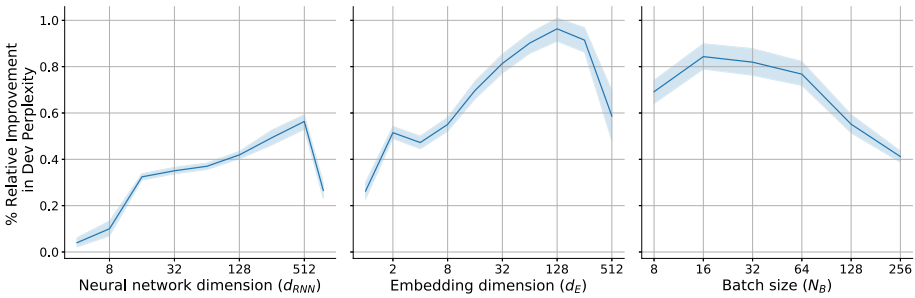

Fig. 2. Relative improvement in development set perplexity achieved during the successive optimisation of the three hyperparameters: neural network dimension d_{RNN}, embedding dimension d_{E}, and batch size N_{B}. Average improvements are calculated over the four language pairs and the three epochs ($N_{\mathrm{E}} = \{20, 40, 60\}$) at which the 8000 sequences in Phase One of hyperparameter tuning are synthesised.

In Phase Two of our optimization process (Sect. 3.3) the temperature (τ) is optimised for each of the text generation strategies in Table 3. Table 7 presents these optimal temperatures for the English-isiZulu sub-corpus. We see that, of the values considered, $\tau = 1.5$ was optimal for all text generation strategies. We also observe that including all generated text (All_{Text}) reduces perplexity and code-switched perplexity by 6.71% and 9.92% respectively relative to the baseline. When only sequences with language switches are retained (CS_{Text}), both perplexity and code-switched perplexity show greater relative improvements of 8.79% and 17.4%. Prompting increased the proportion of synthesised sequences that contain code-switches from 32.0% (All_{Text}) to 87.6% (All_{Prompt}) and affords further perplexity improvements. Finally, combining both ablation and prompting (CS_{Prompt}) produced the best result of the four strategies and reduces perplexity and code-switched perplexity by 9.58% and 20.1% relative to the baseline. When this optimised LSTM is used to generate a larger dataset (1,000,000 sequences), further gains are achieved.

Table 8 presents the perplexity and code-switched perplexity of the interpolated language model for four temperatures when using the CS_{Prompt} strategy (Row 5 in Table 7). When artificial text is generated without modifying the LSTM distribution, the perplexity of the resulting interpolated language model is 6.23% better than the baseline. For the optimized temperature, this improvement increases to 9.58% for the overall perplexity and a more substantial 18.3% for the code-switched perplexity.

We note that higher temperatures produce lower perplexities and therefore more surprising predictions are more helpful for language modelling. Furthermore, the mixture weight λ shifts towards the synthesized language model the more the synthesized data is optimized. Finally, the standard deviation of the perplexities is two orders of magnitude smaller than the corresponding means for both PP and CPP, indicating a high degree of consistency.

Table 7. Mean ± stdev English-isiZulu development set perplexity (PP) and code-switched perplexity (CPP) over 20 runs for the four text generation strategies in Table 3. Temperature τ and average optimal interpolation weight λ are indicated.

Strategy	τ	λ	PP	CPP
Baseline	–	–	626	3226
All_{Text}	1.5	0.76	584 ± 2.5	2906 ± 56.3
CS_{Text}	1.5	0.73	571 ± 3.1	2666 ± 67.0
All_{Prompt}	1.5	0.73	568 ± 4.1	2628 ± 77.1
CS_{Prompt}	1.5	0.73	566 ± 2.5	2579 ± 61.8
1 Mil	1.5	0.73	559 ± 3.6	2513 ± 57.5

Table 8. Mean ± stdev English-isiZulu development set perplexity (PP) and code-switched perplexity (CPP) over 20 runs when interpolating LM_B with an n-gram language model trained on the data generated using CS_{Prompt} text synthesis for different temperatures τ. The associated average optimal interpolation weight is indicated by λ.

τ	λ	PP	CPP
–	–	626	3226
0.75	0.82	587 ± 3.5	3045 ± 61.9
1.0	0.75	570 ± 2.3	2798 ± 48.7
1.25	0.72	565 ± 3.8	2636 ± 60.9
1.5	0.73	566 ± 2.5	2579 ± 61.8

4.2 Speech Recognition

Table 9 shows the perplexities and word error rates achieved by each of the four augmented language model configurations outlined in Table 5 for all four language pairs.

We see again that the code-switched perplexity (CPP) is much larger than the overall perplexity (PP) for all language pairs. Furthermore, the larger vocabularies (Table 1) of isiZulu and isiXhosa are reflected in the higher perplexities, when compared with Sesotho and Setswana. Finally, we see that the speech recognition error rate at language switches (CSBG) is much higher than the overall word error rate.

Table 9 shows that the language models incorporating the synthesized data (LM_{B+S}) reduce code-switched perplexities by between 6.8% and 19.9% relative to the baseline (LM_B). Furthermore, Table 9 shows that, for all four language pairs, the language models that incorporate synthesized data (either LM_{B+S} or LM_{B+S+M}) achieved improvements over the baseline in terms of overall word error rate (1.76% to 2.83% absolute) and also code-switched bigram error (0.86% to 1.37% absolute). From this we conclude that the synthesized text is able to reliably reduce the confusion around code-switched points. Additionally, we note

Table 9. Test set perplexity (PP), code-switched perplexity (CPP), monolingual perplexity (MPP), word error rate (WER), monolingual English and Bantu word error rates (WER_{ENG}, WER_{BAN}) and code-switched bigram (CSBG) error for the four language model configurations in Table 5.

Pair	Model	PP	CPP	MPP	WER	WER_{ENG}	WER_{BAN}	CSBG
EZ	LM_B	842.4	3637	534.8	41.40	36.56	45.12	62.66
	LM_{B+S}	758.5	**2912**	500.8	42.15	36.67	46.36	**61.52**
	LM_{B+M}	624.9	3550	367.4	**38.39**	31.35	**43.81**	62.67
	LM_{B+S+M}	**597.3**	2954	**365.8**	38.57	**31.29**	44.18	61.69
EX	LM_B	1018.0	5171	624.9	42.53	36.71	46.98	68.29
	LM_{B+S}	979.8	**4818**	611.3	41.83	36.08	**46.23**	**66.92**
	LM_{B+M}	**790.0**	5055	**456.2**	**40.56**	**30.87**	47.98	67.64
	LM_{B+S+M}	793.6	4883	463.5	40.77	31.0	48.24	68.06
ES	LM_B	285.9	1166	209.0	49.56	40.2	**56.95**	66.88
	LM_{B+S}	269.7	1016	202.5	49.49	39.44	57.44	66.53
	LM_{B+M}	**223.1**	1080	**158.9**	**47.14**	**33.98**	57.53	66.69
	LM_{B+S+M}	223.9	**979.4**	163.0	47.23	34.71	57.12	**65.91**
ET	LM_B	224.5	1025	154.3	40.80	31.01	47.23	54.90
	LM_{B+S}	202.3	864.8	142.8	40.24	29.96	47.0	**54.04**
	LM_{B+M}	179.8	963.3	120.6	**38.36**	**26.23**	46.34	54.47
	LM_{B+S+M}	**171.7**	**815.5**	**118.4**	38.56	26.84	**46.28**	54.91

that the reduction in code-switched error rate achieved by our language models incorporating the synthesized data (LM_{B+S}) is not achieved by incorporating only the monolingual data (LM_{B+M}). This emphasises the importance of obtaining more code-switched data, which we achieved here through synthesis.

Language models incorporating only the out-of-domain monolingual data (LM_{B+M}) substantially lower the overall and the monolingual perplexities (by between 19.9% and 31.3%) relative to the baseline. These language models also improve the speech recognition performance, outperforming the baseline by between 1.97% and 3.01% absolute in terms of word error rate. The strong correspondence between decreases in monolingual and overall perplexities and word error rate is due to the fact that, despite code switches, most speech remains monolingual. As expected, however, the additional monolingual data has little effect on the code-switched perplexity.

We also note that, for all four language pairs, the language models that incorporate only monolingual data (LM_{B+M}) result in the best overall word error rate, marginally outperforming the language models that also incorporate synthesized data (LM_{B+S+M}) by between 0.09% and 0.21% absolute. However, the columns WER_{ENG} and WER_{BAN} of Table 9 show that this is due mostly to an improvement in the English word error rate (absolute improvements of between 4.78% and 6.22%). Importantly, this is often at the expense of the

Bantu word error rate. Additionally the inclusion of only the monolingual data only marginally affects the recognition error at code-switches (between 0.01% worse and 0.65% better). In terms of code-switched bigram error, the language models which incorporated only the synthesized data outperformed the baseline language models and also the language models which incorporated only the monolingual data for all four language pairs.

5 Conclusion

We have presented a strategy that optimises an LSTM specifically for the purpose of generating code-switched utterances that can be used for language model augmentation. We find that, while our previous LSTMs that are not optimised in this way did not afford improvements, the optimised models are able to generate text the leads to consistent and substantial perplexity and word error rate improvements in all four considered language pairs, especially at language switches. We also see that, although out-of-domain but monolingual data does produce slightly better average speech recognition performance, this improvement is primarily seen for English and is at the expense of the performance in the four under-resourced languages as well as at language switches. We conclude that the hyperparameter tuning, ablation and prompting are effective techniques for improving the speech recognition accuracy at language switches severely under-resourced code-switched datasets.

Acknowledgments. We would like to thank the Council for Scientific and Industrial Research (CSIR), Department of Science and Technology, South Africa for providing access to their CHPC cluster. We gratefully acknowledge the support of Telkom South Africa.

References

1. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous systems. White Paper (2015)
2. Biswas, A., van der Westhuizen, E., Niesler, T., de Wet, F.: Improving ASR for code-switched speech in under-resourced languages using out-of-domain data. In: Proceedings of the 6th International Workshop on Spoken Language Technologies for Under-Resourced Languages (SLTU), Gurugram, India (2018)
3. Biswas, A., Yilmaz, E., de Wet, F., van der Westhuizen, E., Niesler, T.: Semi-supervised development of ASR systems for multilingual code-switched speech in under-resourced languages. In: Proceedings of the 12th Language Resources and Evaluation Conference (LREC), Marseille, France (2020)
4. Biswas, A., Yılmaz, E., de Wet, F., van der Westhuizen, E., Niesler, T.: Semi-supervised acoustic model training for five-lingual code-switched ASR. In: Proceedings of Interspeech, Graz, Austria (2019)

5. Chang, C.T., Chuang, S.P., Lee, H.Y.: Code-switching sentence generation by generative adversarial networks and its application to data augmentation. In: Proceedings of Interspeech, Graz, Austria (2019)
6. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar (2014)
7. Deuchar, M.: Welsh-English code-switching and the matrix language frame model. Lingua **116**(11), 1986–2011 (2006)
8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
9. Gao, Y., Feng, J., Liu, Y., Hou, L., Pan, X., Ma, Y.: Code-switching sentence generation by BERT and generative adversarial networks. In: Proceedings of Interspeech, Graz, Austria (2019)
10. Garg, S., Parekh, T., Jyothi, P.: Code-switched language models using dual RNNs and same-source pretraining. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Brussels, Belgium (2018)
11. Graves, A.: Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850 (2013)
12. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
13. Hu, X., Zhang, Q., Yang, L., Gu, B., Xu, X.: Data augmentation for code-switch language modeling by fusing multiple text generation methods. In: Proceedings of Interspeech, Shanghai, China (2020)
14. Jozefowicz, R., Zaremba, W., Sutskever, I.: An empirical exploration of recurrent network architectures. In: Proceedings of the International Conference on Machine Learning (2015)
15. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
16. Lee, G., Yue, X., Li, H.: Linguistically motivated parallel data augmentation for code-switch language modeling. In: Proceedings of Interspeech, Graz, Austria (2019)
17. Myers-Scotton, C.: Duelling Languages: Grammatical Structure in Codeswitching. Oxford University Press, Oxford (1997)
18. Povey, D., et al.: The Kaldi speech recognition toolkit. In: Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), Hawaii, USA (2011)
19. Stolcke, A.: SRILM-an extensible language modeling toolkit. In: Proceedings of the Seventh International Conference on Spoken Language Processing (ICSLP), Colorado, USA (2002)
20. Taneja, K., Guha, S., Jyothi, P., Abraham, B.: Exploiting monolingual speech corpora for code-mixed speech recognition. In: Proceedings of Interspeech, Graz, Austria (2019)
21. van der Westhuizen, E., Niesler, T.: A first South African corpus of multilingual code-switched soap opera speech. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC). European Language Resources Association (ELRA), Miyazaki (2018)
22. van der Westhuizen, E., Niesler, T.: Automatic speech recognition of English-isiZulu code-switched speech from South African soap operas. Procedia Comput. Sci. **81**, 121–127 (2016). 5th Workshop on Spoken Language Technologies for Under-resourced languages (SLTU), Yogyakarta, Indonesia

23. van der Westhuizen, E., Niesler, T.R.: Synthesised bigrams using word embeddings for code-switched ASR of four South African language pairs. Comput. Speech Lang. **54**, 151–175 (2019)
24. Yılmaz, E., van den Heuvel, H., van Leeuwen, D.: Acoustic and textual data augmentation for improved ASR of code-switching speech. In: Proceedings of Interspeech, Hyderabad, India (2018)