Low-Resource Language Modelling of South African Languages

Stuart Mesham **Luc Hayward** **Jared Shapiro** **Jan Buys**
Department of Computer Science
University of Cape Town, South Africa
{MSHSTU001,HYWLUC001,SHPJAR002}@myuct.ac.za, jbuys@cs.uct.ac.za

Abstract

Language models are the foundation of current neural network-based models for natural language understanding and generation. However, research on the intrinsic performance of language models on African languages has been extremely limited, which is made more challenging by the lack of large or standardised training and evaluation sets that exist for English and other high-resource languages. In this paper, we evaluate the performance of open-vocabulary language models on low-resource South African languages, using byte-pair encoding to handle the rich morphology of these languages. We evaluate different variants of n-gram models, feedforward neural networks, recurrent neural networks (RNNs), and Transformers on small-scale datasets. Overall, well-regularized RNNs give the best performance across two isiZulu and one Sepedi datasets. Multilingual training further improve performance on these datasets. We hope that this research will open new avenues for research into multilingual and low-resource language modelling for African languages.

1 Introduction

Language modelling has applications in many areas of NLP including machine translation, information retrieval, voice recognition and question answering (Wu et al., 2016; Franz and Milch, 2002; Chavula and Suleman, 2016; Ndaba et al., 2016; Kumar et al., 2016). Improvements in language modelling have resulted in improved model performance in the above tasks, making language modelling a valuable area of study. High resource languages have enjoyed substantial improvements in language modelling performance in recent years due to large neural models such as GPT-2, BERT and XLNet (Radford et al., 2019; Devlin et al., 2019; Yang et al., 2019). However, most African languages are low-resource, and the limited availability of high-quality training data makes training large language models challenging.

Ubusuku obuhle namaphupho amamnandi!
Ubu _suku obu _hle nama _phupho ama _mnandi !

Robalang gabotse
R _o _ba _la _ng gabotse

Figure 1: Example sentences and their BPE tokenizations in isiZulu (top) and Sepedi (bottom). The tokenizers use BPE vocabulary sizes of 8000 and 2000 respectively.

In this paper we focus on South African Benue-Congo languages, which are more resourced than most other Benue-Congo languages, but still clearly low-resourced.[1] The two groups of South African languages with the largest number of total speakers are the Nguni and Sotho-Tswana groups of closely-related languages. In South Africa these languages represent 43.3% and 24.7% of speakers respectively (Africa, 2012). In our data sources, the isiZulu and Sepedi languages had the largest amounts of text available, respectively, within these language groups.

In addition to the lack of large amounts of high quality data, Benue-Congo languages are typologically[2] very different from the Indo-European languages most widely studied for language modelling. Even in large multilingual studies, African languages are usually underrepresented if included at all. Benue-Congo languages are agglutinative and morphologically rich (Pretorius and Bosch, 2009): Most words are made up by combination of smaller morphological units, grammatical relations (such

[1] The Benue-Congo languages is a subdivision of the Niger-Congo language family. Most Benue-Congo languages are part of what linguists refer to as the Bantu or Bantoid subfamilies.

[2] Typology refers to the linguistic properties and characterization of a language.

as subject or object) are indicated by changes in the words rather than the relative position of words in the sentence, and all nouns belong to one of a large number of noun classes which governs the choice of many morphemes. This leads to potentially very large and sparse word-level vocabulary, even though individual morphemes or sub-words many be more frequent in a corpus (as they are used in many different words).

This paper examines the application of n-gram models (Chen and Goodman, 1999), Feed-forward neural networks (FFNNs) (Bengio et al., 2003), Recurrent neural networks including Long Short Term Memory (LSTMs; Hochreiter and Schmidhuber, 1997) and Transformer (Vaswani et al., 2017) models on isiZulu and Sepedi. We use byte pair encoding (BPE; Sennrich et al., 2016) to control the vocabulary size and to enable open-vocabulary language modelling (see Figure 1), making the choice of vocabulary size a hyperparameter of the models.

Our results show that the relative performance of the different model classes is similar to what have been found in previous work on small-scale language modelling in English and other languages. Well-regularized RNNs, the AWD-LSTM (Merity et al., 2017) and QRNN (Bradbury et al., 2017), have the best overall performance, outperforming the Transformer. The n-gram, FFNN and baseline LSTM models performed worse across all datasets. We also perform an evaluation of multilingual training, showing that training on text from multiple related languages improves performance without any modifications to the model architecture. The benefits can be seen using text from either the same language group or a different but related language group, despite orthographic differences. Code and trained models can be found at `https://github.com/StuartMesham/low_resource_lm`.

2 Background

A language model assigns a probability $P(W_1^n)$ to a sequence of n words $W_1^n = w_1, ..., w_n$. The probability is usually decomposed using the chain rule to predict the words one at a time (from left to right) by assigning a probability to each word for following the given context (Jurafsky and Martin, 2020):

$$P(W_1^n) = \prod_{k=2}^{n} P(w_k|W_1^{k-1}). \quad (1)$$

2.1 Sub-word Tokenization

Language models traditionally estimate the next word probability as a distribution over a fixed vocabulary, where the input text has been tokenized into words, and all words outside the vocabulary replaced with a special *unknown* token. South African Benue-Congo languages are highly agglutinative, making whole-word tokenization suboptimal for language modelling due to potentially large vocabulary sizes and subsequent data sparsity. In contrast, character-level tokenization requires the model to learn to model very long sequences. To better represent the structure of the languages, we use byte-pair encoding (Gage, 1994; Sennrich et al., 2016) to break words into sub-word units based on their frequency. Language modelling with BPE has previously been shown to perform competitively for open-vocabulary language modelling (Mielke and Eisner, 2019).

Byte-pair encoding is a compression algorithm which has been adapted for sub-word tokenization. The algorithm starts with character-level tokens and finds pairs of adjacent tokens which occur most frequently. These token pairs are replaced with single tokens containing the concatenation of the characters in each token. This process is repeated until a desired vocabulary size is reached (Sennrich et al., 2016). To ensure fair model evaluation, we train BPE tokenizers using only the training sets. Example BPE tokenizations in isiZulu and Sepedi are shown in Figure 1.

2.2 Evaluation

The quality of a language model can be evaluated either extrinsically or intrinsically. Extrinsic evaluation measures a model's usefulness in some downstream task such as speech recognition or machine translation whereas intrinsic evaluation uses statistical measures to assess a model's quality. In this paper we focus on intrinsic evaluation metrics related to cross-entropy and perplexity.

In information theory, entropy represents the average number of units of information produced per observation (Shannon, 1948). The cross-entropy of a language model on a given sample of text W_1^n is estimated as

$$H(W_1^n) = -\frac{1}{n} \log_2 P(W_1^n), \quad (2)$$

with the units of information being bits due to the log base 2 (Jurafsky and Martin, 2020). The more accurately the model approximates the true distribution of the language, the lower the cross-entropy. Language models with a fixed vocabulary are usually evaluated based on perplexity, which is computed as $2^{H(W_1^n)}$. However, closed-vocabulary language models have to set the size of the vocabulary and treat all other words as unknown. Consequently, perplexity cannot be compared directly across models with different vocabularies.

In this paper we are studying open-vocabulary models, and we want the choice of tokenization and vocabulary to be a modelling choice. This necessitates an evaluation metric which is independent of the tokenization.

As evaluation metric we use bits per character (BPC), a measure of cross-entropy which is normalised by the character length of the text and is therefore independent of the tokenization. The BPC of a model on a test set W_1^n is calculated as

$$\mathrm{BPC}(W_1^n) = \frac{n}{c} H(W_1^n), \tag{3}$$

where the text consists of c characters.

2.3 Models

2.3.1 n-gram Models

n-gram language models make the Markov assumption of restricting the context for predicting the next word to the last $n-1$ words (Jurafsky and Martin, 2020). Traditional n-grams are based on various smoothing methods, of which modified Knesser-Ney smoothing has been shown to lead to the best performance in general (Kneser and Ney, 1995; Chen and Goodman, 1999). Sparsity increases as the n-gram size increases, which leads to practical limits on the size of n that is used.

2.3.2 Feedforward Neural Networks

The first neural network-based language models were based on feedforward neural networks (FFNNs), which also make the Markov assumption, and are therefore effectively neuralized n-gram models (Bengio et al., 2003). One of the key advantages of neural language models over n-grams is that word embeddings allow them to generalise better, as words with similar meanings or grammatical functions will have similar embeddings (Mikolov et al., 2013).

The first layer of an FFNN takes the concatenation of the context word embeddings as input. The embedding layer is learned jointly with the rest of the model and weight-tied to the output layer, following standard practice in RNN-based language modelling. We use a rectified linear unit as nonlinearity.

2.3.3 LSTMs

LSTMs (Hochreiter and Schmidhuber, 1997) are a widely used variant of the standard RNN architecture allowing for longer term dependencies to be modelled more effectively by using a number of gates along with a memory vector in the recurrent cell. The gates and the memory vector enable information to pass more effectively across time steps. We use a Basic-LSTM model as a baseline for the more complex AWD-LSTM and QRNN models (see below).

This model is regularized using dropout, which temporarily hides a random subset of neurons during each training step (Srivastava et al., 2014). This adds noise and prevents the model from being overly reliant on any particular neuron. However, dropout in RNN models cannot be applied between time steps on the recurrent connection as it inhibits the model's ability to retain long term dependencies, so the standard approach is to apply dropout only on the input and output connections (Zaremba et al., 2015). The Basic-LSTM baseline does not use the more complex regularization and optimization techniques used by the other models.

2.3.4 AWD-LSTM

The AWD-LSTM model (Merity et al., 2018b) is used widely for language modelling and forms the bases of the current state-of-the-art language modelling on small English datasets without dynamic evaluation (Takase et al., 2018). In order to enable a fair comparison across models we are not using a continuous cache pointer (Grave et al., 2017) or dynamic evaluation.

The AWD-LSTM uses a number of improved regularization and optimization techniques. Regularization is particularly important in low-resource settings. DropConnect (Wan et al., 2013) is a form of dropout on the hidden-to-hidden weights.[3] Variational dropout (Gal and Ghahramani, 2016) generates a dropout mask once which is then used over the entire forward and backward pass, rather than resampling at every timestep. The AWD-LSTM

[3]This method is particularly useful as it is applied once to the weight matrices before the forward and backward pass, allowing the use of black box RNN implementations such as NVIDIA's cuDNN LSTM which can be many times faster due to hardware optimisations (Merity et al., 2018b).

model uses a combination of DropConnect for the hidden-to-hidden transitions within the LSTM and variational dropout over the inputs and outputs. Other tehchniques used include using variable length backpropagation sequences, word dropout (masking entire word embeddings), and L1 and L2 regularisation.

2.3.5 Quasi-Recurrent Neural Networks

Quasi-Recurrent Neural Networks (QRNNs) (Bradbury et al., 2017) is a modification of RNNs that parallizes parts of the RNN computation and obtained similar or even slightly better performance than the AWD-LSTM on some English datasets (Merity et al., 2018a). The QRNN applies convolutional layers on the input, followed by an recurrent pooling function resembling LSTM gating. This significantly increases training speed compared to LSTMs of similar sizes.

2.3.6 Transformers

The Transformer (Vaswani et al., 2017) presents another approach to speeding up sequential processing over RNNs by relying entirely on attention mechanisms (Bahdanau et al., 2015) instead of recurrent connections for propagating information across time steps. An attention mechanism can process all the input embeddings for a (fixed-length) sequence simultaneously and selectively weight certain features based on a learned function.

The original Transformer model was used for translation and has an encoder-decoder structure (Vaswani et al., 2017). For the task of language modelling, only the decoder architecture is used (Liu et al., 2018). We follow the architecture used by GPT-2 (Radford et al., 2019). A learned positional embedding is added to each input token embedding. Multiple layers, each including an attention and a feedforward sub-layer, are stacked to create the larger model that can propagate information more efficiently across time steps. In each attention sub-layer multiple attention mechanisms are used to extract features; this strategy is termed multi-headed self-attention. Finally, a residual connection and layer normalisation is applied over each sub-layer. To regularise the Transformer models we use dropout on all weights of the model.

3 Experimental Setup

3.1 Datasets

We focus on language modelling for isiZulu and Sepedi, but we processed data for all 11 non-European official South African languages, and use the other languages' data for multilingual training (Section 5). We use two dataset sources:

Corpus	**Words**	
	Training	Valid/Test
NCHLT (isiZulu)	978.6	122.3
Isolezwe (isiZulu)	940.2	117.5
NCHLT (Sepedi)	1357.3	169.7

Table 1: Dataset sizes, reported in thousands of words, after preprocessing. The validation and test sets of each corpus are approximately equal in size.

NCHLT: We use the corpora from the National Centre for Human Language Technology (NCHLT) Text project (Eiselen and Puttkammer, 2014) made available by the South African Centre for Digital Language Resources (SADiLaR).[4] Monolingual text corpora are available for all 11 of South Africa's official languages. We processed the corpora for the Nguni languages (isiZulu, Siswati, isiNdebele and isiXhosa) and the Sotho-Tswana languages (Sesotho, Sepedi, Setswana), as well as Xitsonga and Tshivenda, the other two Benue-Congo languages. A significant proportion of these texts were scraped from governmental websites. The corpora range in size from 1 to 3 million tokens. Sepedi and isiZulu have the largest datasets in their respective language groups.

Isolezwe: News articles from the isiZulu Isolezwe newspaper, one of the largest daily African language newspapers in South Africa, have been scraped and consolidated by the Newstools initiative.[5] This is the largest publicly-available newspaper corpus among the languages we are considering that we are aware of. The dataset has a similar size to the NCHLT isiZulu corpus but provides a second evaluation domain.

We performed a number of data proprocessing and normalization steps. We removed instances of English, HTML and Javascript lines, and other repetitive or erroneous data, as these would not naturally be found in general language. Each dataset was split into a training, validation and test set using an 80% / 10% /10% split. The splits were done using sequential blocks to preserve the order of the sentences. Table 1 compares the dataset sizes.

[4] Datasets are available at `https://repo.sadilar.org/handle/20.500.12185/7`

[5] Available at `https://github.com/newstools`

3.2 Model Implementation and Optimization

The BPE preprocessing for all models uses the HuggingFace tokenizers library.[6]

3.2.1 n-gram Models

We use an n-gram language model with modified Kneser-Ney (Chen and Goodman, 1999) smoothing, as implemented in KenLM.[7] We tuned the models by testing BPE vocabulary sizes ranging from 100 to 10000 and n-gram orders from 2 to 6. The isiZulu and Sepedi models performed best with BPE vocabulary sizes of 500 and 2000 respectively. For all datasets, an n-gram order of 6 yielded the best performance.

3.2.2 Feedforward Neural Networks

We implemented a feed-forward neural network (FFNN) language model so that it can be trained in a similar manner to RNN and Transformer language models. The training data is divided into chunks of 64 tokens and batched to enable parallel processing. We follow the optimization and regularization setup of the FFNN baseline used by Chiu and Rush (2020). We use of a learning rate decay schedule where the learning rate is multiplied by 0.25 after each epoch if the validation loss does not improve. The models were trained for 50 epochs with a batch size of 32 and an AdamW weight decay of 0.01. Both word embeddings and hidden layers had a size of 500.

Using grid search, we evaluated BPE vocabulary sizes 1000 and 2000 to 10 000 with an interval size of 2000, n-gram orders {2, 4, 6}, word embedding and hidden layer sizes in the range {500, 2500} with an interval of 250, dropout rates of {0.3, 0.5} and {2, 4, 6} hidden layers.

For both NCHLT isiZulu and NCHLT Sepedi a BPE vocabulary size of 8000 yielded the best performance, and on Isolezwe 10 000 performed best. For both Isolezwe and NCHLT isiZulu, an n-gram order of 2 performed best and for NCHLT Sepedi an order of 4. We were unable to find a fully satisfactory explanation of why the FFNN did not perform better with a higher n-gram orders.

3.2.3 LSTMs

We use the PyTorch implementation of the AWD-LSTM (Merity et al., 2018b).[8] We took the hyper-

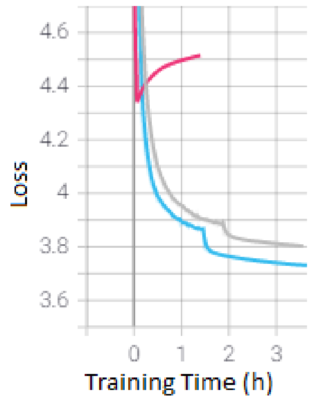

Figure 2: The validation loss of the basic LSTM (pink), AWD-LSTM (grey) and QRNN (blue) while training on the NCHLT isiZulu dataset.

parameters of Merity et al. (2018b) on the word-level WikiText 2 dataset as the starting point for tuning our models, as its size is comparable to our dataset. We performed a partial grid search over the embedding size {400, 800}, hidden layer size {1150, 1200, 1550}, number of layers {1, 2, 3, 4}, learning rate {5, 10, 30}, batch size {40, 80}, vocab size {2500, 5000, 7500, 10000} as well as dropout rate {0 - 0.7} and weight drop {0 - 0.5} (both in increments of 0.1) and L1/L2 regularisation values {0, 1, 2}. Model development was primarily done on the isiZulu NCHLT corpus. Most improvement came from increasing the total model size by either increasing the number of hidden layers or increasing the input embedding size. Changing the BPE vocabulary size did not have a significant effect on performance. The Basic LSTM was tuned similarly, excluding the regularization techniques it does not implement.

3.2.4 QRNN

The QRNN is also implemented in the AWD-LSTM packages. We tuned the embedding size, vocabulary size, number of hidden layers and batch size, using similar ranges as for the AWD-LSTM. The best QRNNs used an embedding size of 800, hidden layer sizes of 1550, and 4 hidden layers.

Figure 2 shows how the validation loss changes while the RNN-based models (Basic LSTM, AWD-LSTM, and QRNN) train on the NCHLT isiZulu corpus. The plot shows how the QRNN's loss decreases faster than that of the AWD-LSTM time. The Basic LSTM initially trains faster, but then overfits drastically.

[6] https://github.com/huggingface/tokenizers
[7] Available at https://github.com/kpu/kenlm
[8] Available at https://github.com/salesforce/awd-lstm-lm

3.2.5 Transformers

We used the GPT-2 (Radford et al., 2019) PyTorch implementation provided by the open-source HuggingFace transformers library.[9] The training data was fed to the model in blocks of 128 consecutive tokens with a batch size of 32, created using a sliding window over the training data with a stride of 16 tokens. Model evaluation was performed using an input block size of 128 with a stride of 64.

For hyper-parameter tuning, models were trained for up to 200k steps, with evaluation on a validation set every 5k steps. Training was stopped early if the validation loss did not decrease after any four successive evaluations. The model and vocabulary sizes were tuned first with little regularization to ensure that the models had enough capacity to overfit the data. Increasing amounts of regularization were then applied until the model no longer overfit the data.

We used 8 hidden layers and 8 attention heads. Preliminary experiments showed that the model was relatively insensitive to the number of hidden layers and the number of attention heads. We used an initial learning rate of 10^{-4} with a learning rate schedule that linearly decreases to 0 over the course of the training. Across all 3 corpora, the best performing models had a hidden layer size of 256, a dropout probability of 0.3 and a weight decay of 0.2. The isiZulu and Sepedi models performed best with BPE vocabulary sizes of 8000 and 2000 respectively.

4 Results and Discussion

4.1 Results

All the test set results are given in Table 2. The n-gram and FFNN language models performed fairly similarly to each other across the datasets and languages, even though the FFNNs used smaller n-gram orders. On the isiZulu datasets, the FFNN performed slightly better than the n-gram models, while on the Sepedi dataset the n-gram model performed better. On all datasets, we found that the n-gram models tended to perform better with smaller BPE vocabulary sizes, whereas the FFNN models performed better with larger vocabulary sizes.

The performance of the AWD-LSTM and QRNN models was closely matched (within 0.005 BPC) across all datasets with the QRNN slightly outperforming on the two NCHLT datasets, and the AWD-LSTM ahead on the Isolezwe dataset. The basic LSTM under-performed the others substantially, with performance closer to, or even worse than, that of the n-gram and FFNN models.

The transformer models achieved competitive performance on all datasets, but were outperformed by the QRNN and AWD-LSTM. We hypothesize that the main reason is that these models used more sophisticated regularization techniques that our Transformer implementation did not use. Additionally, the RNNs had more parameters, but the Transformer's performance did not improve with more parameters in our experiments.

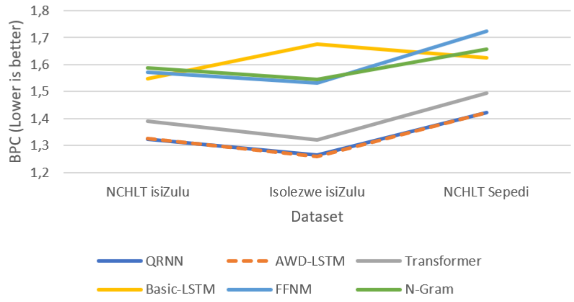

Figure 3: Test set results, plotted to show the relative performance of the models on each of the three datasets (lower is better). The AWD-LSTM and QRNN consistently outperform the other models while within close margin of each other, followed by the Transformer, while the n-gram, FFNN and Basic-LSTM perform substantially worse.

4.2 Discussion

The results show that the relative performance of the models is similar to language modelling results previously reported on widely used PTB and WikiText2 English datasets (Merity et al., 2017), which are comparable in size to our corpora. Regarding the performance of the Transformer, it has been reported that a modified Transformer architecture with segment-level recurrence can obtain similar results to the AWD-LSTM when fine-tuned using the same sophisticated regularization techniques (Dai et al., 2019), but other researchers have struggled to reproduce these results independently.[10]

We found that the relative performance of the language models was similar across the three datasets (Figure 3). This supports the hypothesis that the same models would likely perform well across all the languages in the Nguni and Sotho-Tswana language groups. The AWD-LSTM and QRNN mod-

[9] https://huggingface.co/transformers/

[10] https://twitter.com/srush_nlp/status/1245825437240102913

Model	NCHLT (isiZulu)			Isolezwe (isiZulu)			NCHLT (Sepedi)		
	Params	Vocab	BPC	Params	Vocab	BPC	Params	Vocab	BPC
n-gram	7.5M	500	1.588	6.9M	500	1.544	5.7M	2000	1.656
FFNN	4.7M	8000	1.572	5.7M	1000	1.532	5.1M	8000	1.723
Basic-LSTM	3.3M	5000	1.548	3.3M	5000	1.677	3.3M	5000	1.625
AWD-LSTM	29.8M	5000	1.325	29.8M	5000	**1.259**	29.8M	5000	1.421
QRNN	29.5M	10000	**1.323**	29.5M	10000	1.264	29.5M	5000	**1.421**
Transformer	8.6M	8000	1.391	8.6M	8000	1.320	7.1M	2000	1.495

Table 2: Language modelling results on the isiZulu and Sepedi corpora, reported as bits-per-character (BPC). The BPE vocabulary size and number of parameters of each model are also reported.

els were consistently close in performance, followed by the transformer model across all datasets. The remaining n-gram, FFNN and Basic-LSTM models had different relative performances on the datasets with no consistent pattern, although the n-gram and FFNN are closer to each other. The poor performance of the n-gram and FFNN models represents a trade-off between training time and model performance. If training time was a factor, reduced performance could be accepted in order to produce models more quickly. The n-gram models are also much faster when queried in downstream applications.

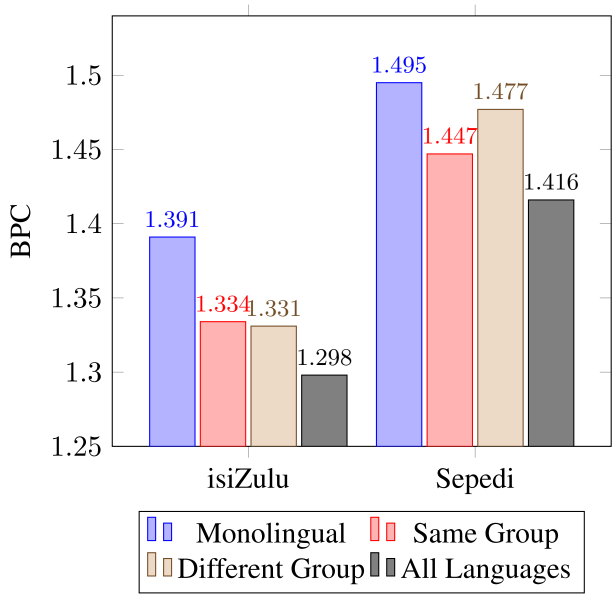

Figure 4: Multilingual language modelling results, reported as bits-per-character (BPC), evaluated on the isiZulu and Sepedi test sets. Models were trained on the target language (Monolingual), and additionally also on multiple languages in the same language group (Nguni and Sotho-Tswana, respectively), languages from the other language group, or on text from all 9 non-European official South African languages.

5 Multilingual Models

As an additional experiment, we investigate the potential for multilingual language modelling by concatenating training data from multiple languages and evaluating on the same target languages as before. For practical reasons, we only train Transformer models for this experiment. We use the NCHLT corpora as they provide text in the same domain across all South African languages.

We train models in a number of different settings. In particular, we were interested in comparing the effect of training on additional languages from the same language group (isiZulu: all Nguni languages; Sepedi: all Sotho-Tswana languages) compared to training on languages from the other language group (isiZulu: Sotho-Tswana languages; Sepedi: Nguni languages). Finally, we also evaluated a model trained on all 9 Benue-Congo South African languages in the NCHLT corpus. Model hyperparameters were tuned separately in each instance.

The results are shown in Figure 4. For both target languages, concatenating training data from other Benue-Congo languages improves performance. In general, training on more languages improves performance regardless of the language group. In the case of Sepedi as target language, concatenating the other Sotho-Tswana languages yields a greater performance improvement than concatenating Nguni languages. On the other hand, for isiZulu the results of including additional data from the same or the other langauge family were similar. For both isiZulu and Sepedi models, the best performance is obtained by concatenating data from all languages. We hypothesize that transfer may be more effective from disjunctively written languages

(Sotho-Tswana) to conjunctively written languages (Nguni) than the other way around, but this needs to be investigated further. Our results suggest that the use of data from multiple languages is a promising future direction for modelling South African languages.

6 Conclusions

The experiments conducted in this paper demonstrated that improved regularization techniques and model architectures developed on relatively small English datasets also improves language modelling performance when applied to African languages such as isiZulu and Sepedi. The AWD-LSTM and QRNN performed notably better than the other models. As expected, n-grams and FFNNs, as well as the Basic LSTM, underperformed the more advanced models. However, the stronger models are computationally more expensive. Our results suggest that further improvements in RNN- and Transformer-based language modelling would likely be directly applicable to low-resource African languages. Additionally, we showed that BPE is an effective method for open vocabulary language modelling across multiple models, effectively accounting for the large (word-level) vocabulary sizes of agglutinative African Languages. Finally, we showed that multilingual language modelling is a promising direction for future research, as many African languages occur in groups of closely related languages which might benefit from such an approach.

Acknowledgments

This work is based on research supported in part by the National Research Foundation of South Africa (Grant Number: 129850) and the South African Centre for High Performance Computing.

References

Statistics South Africa. 2012. *Census 2011, Census in Brief*. Statistics South Africa, Pretoria.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2017. Quasi-Recurrent Neural Networks. In *International Conference on Learning Representations*.

Catherine Chavula and Hussein Suleman. 2016. Assessing the impact of vocabulary similarity on multilingual information retrieval for bantu languages. In *Proceedings of the 8th Annual Meeting of the Forum on Information Retrieval Evaluation*, FIRE '16, page 16–23, New York, NY, USA. Association for Computing Machinery.

Stanley F Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394.

Justin Chiu and Alexander Rush. 2020. Scaling hidden Markov language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1341–1349, Online. Association for Computational Linguistics.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Roald Eiselen and Martin Puttkammer. 2014. Developing text resources for ten south African languages. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3698–3703, Reykjavik, Iceland. European Language Resources Association (ELRA).

Alexander Franz and Brian Milch. 2002. Searching the web by voice. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 2*, COLING '02, page 1–5, USA. Association for Computational Linguistics.

Philip Gage. 1994. A new algorithm for data compression. *C Users Journal*, 12(2):23–38.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 1027–1035, Red Hook, NY, USA. Curran Associates Inc.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2017. Improving neural language models with a continuous cache. In *International Conference on Learning Representations*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Daniel Jurafsky and James H Martin. 2020. N-gram Language Models. In *Speech and Language Processing*, 3 edition, chapter 3. Online Draft.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for M-gram language modeling. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, volume 1, pages 181–184. IEEE.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1378–1387. JMLR.org.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018a. An Analysis of Neural Language Modeling at Multiple Scales. *CoRR*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018b. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations*.

Stephen Merity, Bryan Mccann, and Richard Socher. 2017. Revisiting Activation Regularization for Language RNNs. *CoRR*.

Sebastian J Mielke and Jason Eisner. 2019. Spell once, summon anywhere: A two-level open-vocabulary language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6843–6850.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

B. Ndaba, H. Suleman, C. M. Keet, and L. Khumalo. 2016. The effects of a corpus on isizulu spellcheckers based on n-grams. In *2016 IST-Africa Week Conference*, pages 1–10.

Laurette Pretorius and Sonja Bosch. 2009. Exploiting cross-linguistic similarities in Zulu and Xhosa computational morphology. In *Proceedings of the First Workshop on Language Technologies for African Languages*, pages 96–103, Athens, Greece. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Technical report, OpenAI.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

C. E. Shannon. 1948. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3):379–423.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Sho Takase, Jun Suzuki, and Masaaki Nagata. 2018. Direct output connection for a high-rank language model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4599–4609, Brussels, Belgium. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pages 1058–1066.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32, pages 5753–5763. Curran Associates, Inc.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2015. Recurrent Neural Network Regularization . *CoRR*.