Low-Resource Language Modelling of South African Languages

Stuart Mesham, Luc Hayward, Jared Shapiro, and Jan Buys

Department of Computer Science, University of Cape Town, 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, and 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 improves performance on these datasets. We hope that this work will open new avenues for research into multilingual and low-resource language modelling for African languages.

Keywords: natural language processing · language modelling · South African languages · multilingual models · recurrent neural networks · transformers

1 Introduction

Language modelling has applications in many areas of NLP including machine translation [33], information retrieval [4], speech recognition [10] and question answering [17]. 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 [26], BERT [8] and XLNet [34]. However, most African languages are low-resource, and the limited availability of high-quality training data makes training large language models challenging.

In this paper we focus on South African Benue-Congo languages, which are better resourced than most other Benue-Congo languages, but still clearly low-resource.[1] The two groups of South African languages with the largest number

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

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

Robalang gabotse
R _o _ba _la _ng gabotse

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

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 [18]. 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 [25]: Most words are formed out of a combination of smaller morphological units; grammatical relations (such 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 may be more frequent in a corpus (as they are used in many different words).

This paper examines the application of n-gram models [5], feed-forward neural networks (FFNNs) [2], recurrent neural networks including Long Short Term Memory (LSTMs) [14] and Transformer [31] models on isiZulu and Sepedi. We use byte pair encoding (BPE) [27] to control the vocabulary size and to enable open-vocabulary language modelling (see Figure 1), making the choice of vocabulary size a hyperparameter.

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 [22] and QRNN [3], 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. The code, data processing scripts, and trained versions of all our models can be found at `https://github.com/StuartMesham/low_resource_lm`.

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

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 [15]:

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

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 sub-optimal 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 [11], [27] 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 [23].

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 [27]. 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 [28]. The cross-entropy of a language model on a given sample of text W_1^n is estimated as

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

with the units of information being bits due to the log base 2 [15]. 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

***n*-gram Models** n-gram language models make the Markov assumption of restricting the conditioning context for predicting the next word to the last $n-1$ words [15]. Traditional n-grams are based on various smoothing methods, of which modified Kneser-Ney smoothing [16] has been shown to lead to the best performance in general [5]. Sparsity increases as the n-gram size increases, which leads to practical limits on the size of n that is used.

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 [2]. 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 [24].

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 non-linearity.

LSTMs LSTMs [14] are a widely used variant of the standard recurrent neural network (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. LSTMs generally perform as well or better than Gated Recurrent Units (GRUs) in language modelling, so we do not consider GRUs or other gated RNN variants here. 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 [29]. 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 [35]. The Basic-LSTM baseline does not use the more complex regularization and optimization techniques used by the other models.

AWD-LSTM The AWD-LSTM model [21] is used widely for language modelling and forms the basis of the current state-of-the-art language modelling on small English datasets without dynamic evaluation [30]. In order to enable a fair comparison across models we do not using a continuous cache pointer [13] or dynamic evaluation.

The AWD-LSTM uses a number of improved regularization and optimization techniques. Regularization is particularly important in low-resource settings. DropConnect [32] is a form of dropout on the hidden-to-hidden weights.[3] Variational dropout [12] 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 model uses a combination of DropConnect for the hidden-to-hidden transitions within the LSTM and variational dropout over the inputs and outputs. Other techniques used include using variable length backpropagation sequences, word dropout (masking entire word embeddings), and L1 and L2 regularisation.

Quasi-Recurrent Neural Networks The Quasi-Recurrent Neural Networks (QRNN) [3] is a modification of RNNs that parallelizes parts of the RNN computation and obtained similar or even slightly better performance than the AWD-LSTM on some English datasets [20]. 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.

Transformers The Transformer [31] presents another approach to speeding up sequential processing over RNNs by relying entirely on attention mechanisms [1] 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 [31]. For the task of language modelling, only the decoder

[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 [21].

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

Corpus	**Tokens (000s)**	
	Training	Valid/Test
NCHLT (isiZulu)	979	122
Isolezwe (isiZulu)	940	117
NCHLT (Sepedi)	1 357	170

architecture is used [19]. We follow the architecture used by GPT-2 [26]. 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:

NCHLT: We use the corpora from the National Centre for Human Language Technology (NCHLT) Text project [9] 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. We use articles published between 2016 and 2020. The dataset has a similar size to the NCHLT isiZulu corpus but provides a second evaluation domain.

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

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

We performed a number of data preprocessing 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.

3.2 Model Implementation and Optimization

The BPE preprocessing for all models uses the HuggingFace tokenizers library.[6] Due to computational constraints we were not able to train and evaluate models across multiple random seeds.

***n*-gram Models** We use an n-gram language model with modified Kneser-Ney [5] 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.

Feedforward Neural Networks We implemented a feed-forward neural network (FFNN) language model in PyTorch so that it can be trained efficiently 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 [6]. We use 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 higher n-gram orders.

LSTMs We use the PyTorch implementation of the AWD-LSTM [21].[8] We tuned its hyperparameters on the word-level WikiText-2 dataset as the starting

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

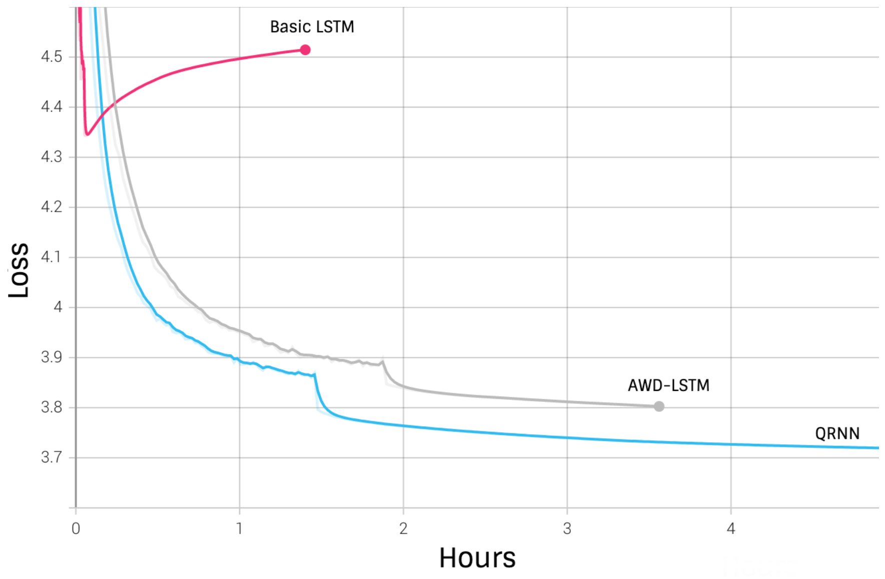

Fig. 2. The validation loss of the basic LSTM, AWD-LSTM and QRNN while training on the NCHLT isiZulu dataset.

point for tuning our models, as the size of that dataset is comparable to ours. 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[9] {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, aside from excluding the regularization techniques it does not use.

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.

[9] Following previous work we start with a large learning rate, which is then reduced during training.

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 given.

	NCHLT (isiZulu)			**Isolezwe (isiZulu)**			**NCHLT (Sepedi)**		
Model	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	10000	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

Transformers We used the GPT-2 [26] PyTorch implementation provided by the open-source HuggingFace *transformers* library.[10] 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 hyperparameter 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

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

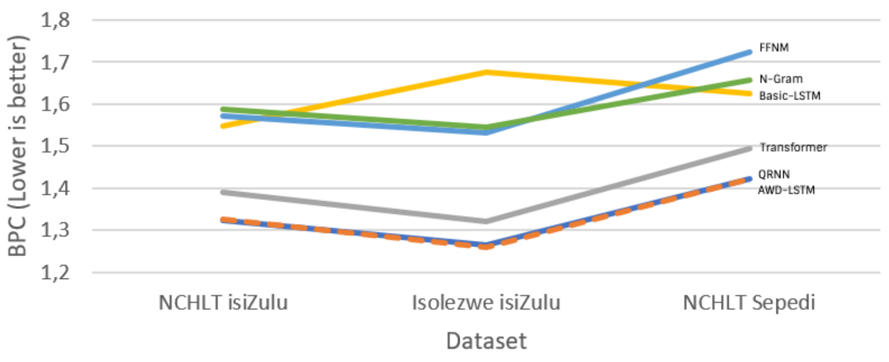

Fig. 3. Test set results (as reported in Table 2), plotted to show the relative performance of the models on each of the three datasets (lower is better). The AWD-LSTM and QRNN consistently outperform the other models while within close margin of each other, followed by the Transformer, while the n-gram, FFNN and Basic-LSTM perform substantially worse.

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.

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 [22], which are comparable in size to our corpora. While BPC results cannot be compared directly across corpora and languages, [23] reported 1.468 for a BPE-based LSTM on WikiText2, and results in the range 1.31 to 1.85 across 7 European languages. Our BPC results are therefore in a similar range to previous work.

Regarding the performance of the Transformer, it has been reported that a modified Transformer architecture with segment-level recurrence can obtain similar results on standard English datasets to the AWD-LSTM when using

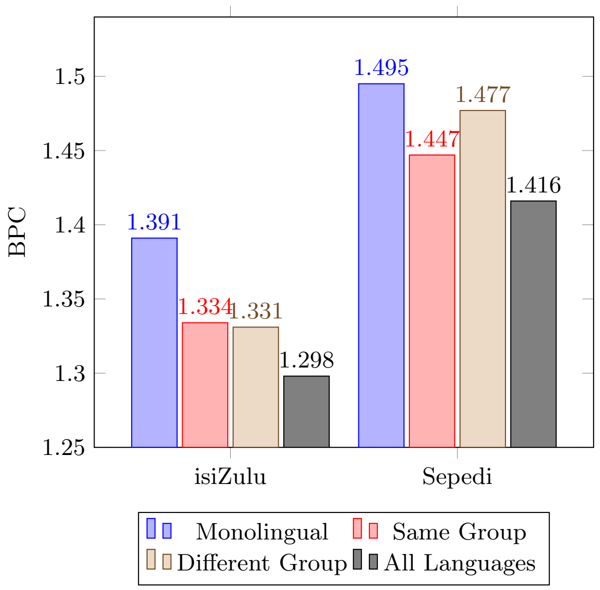

Fig. 4. Multilingual language modelling results, reported as bits-per-character (BPC), evaluated on the isiZulu and Sepedi test sets. Models were trained on the target language (Monolingual), and additionally also on multiple languages in the same language group (Nguni and Sotho-Tswana, respectively),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

the same sophisticated regularization techniques [7], but other researchers have struggled to reproduce these results independently.[11]

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 models 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.

5 Multilingual Models

As an additional experiment, we investigated 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 (all Nguni languages for isiZulu; all Sotho-Tswana languages for Sepedi) 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 using the same methodology as for the monolingual Transformers.

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, under-performed 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,

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

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

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: International Conference on Learning Representations (2015)
2. Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C.: A neural probabilistic language model. Journal of machine learning research **3**(Feb), 1137–1155 (2003)
3. Bradbury, J., Merity, S., Xiong, C., Socher, R.: Quasi-Recurrent Neural Networks. In: International Conference on Learning Representations (2017)
4. Chavula, C., Suleman, H.: 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, p. 16–23. Association for Computing Machinery, New York, NY, USA (2016). DOI 10.1145/3015157.3015160. URL https://doi.org/10.1145/3015157.3015160
5. Chen, S.F., Goodman, J.: An empirical study of smoothing techniques for language modeling. Computer Speech & Language **13**(4), 359–394 (1999)
6. Chiu, J., Rush, A.: Scaling hidden Markov language models. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1341–1349. Association for Computational Linguistics, Online (2020). DOI 10.18653/v1/2020.emnlp-main.103. URL https://aclanthology.org/2020.emnlp-main.103
7. Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., Salakhutdinov, R.: Transformer-XL: Attentive language models beyond a fixed-length context. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 2978–2988. Association for Computational Linguistics, Florence, Italy (2019). DOI 10.18653/v1/P19-1285. URL https://www.aclweb.org/anthology/P19-1285
8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: 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), pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (2019). DOI 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423
9. Eiselen, R., Puttkammer, M.: Developing text resources for ten South African languages. In: Proceedings of the Ninth International Conference on Language Resources and Evaluation, pp. 3698–3703. European Language Resources Association (ELRA), Reykjavik, Iceland (2014). URL http://www.lrec-conf.org/proceedings/lrec2014/pdf/1151_Paper.pdf

10. Franz, A., Milch, B.: Searching the web by voice. In: Proceedings of the 19th International Conference on Computational Linguistics - Volume 2, p. 1–5. Association for Computational Linguistics, USA (2002). DOI 10.3115/1071884.1071887. URL https://doi.org/10.3115/1071884.1071887
11. Gage, P.: A new algorithm for data compression. C Users Journal **12**(2), 23–38 (1994)
12. Gal, Y., Ghahramani, Z.: A theoretically grounded application of dropout in recurrent neural networks. In: Advances in Neural Information Processing Systems, p. 1027–1035. Curran Associates Inc., Red Hook, NY, USA (2016)
13. Grave, E., Joulin, A., Usunier, N.: Improving neural language models with a continuous cache. In: International Conference on Learning Representations (2017)
14. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
15. Jurafsky, D., Martin, J.H.: N-gram Language Models. In: Speech and Language Processing, 3 edn., chap. 3. Online Draft (2020)
16. Kneser, R., Ney, H.: Improved backing-off for M-gram language modeling. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing, vol. 1, pp. 181–184. IEEE (1995). DOI 10.1109/icassp.1995.479394
17. Kumar, A., Irsoy, O., Ondruska, P., Iyyer, M., Bradbury, J., Gulrajani, I., Zhong, V., Paulus, R., Socher, R.: Ask me anything: Dynamic memory networks for natural language processing. In: Proceedings of the 33rd International Conference on Machine Learning, p. 1378–1387. JMLR.org (2016)
18. Lehohla, P.: Census 2011, census in brief: The South africa I know the home I understand (2012)
19. Liu, P.J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., Shazeer, N.: Generating wikipedia by summarizing long sequences. In: International Conference on Learning Representations (2018)
20. Merity, S., Keskar, N.S., Socher, R.: An Analysis of Neural Language Modeling at Multiple Scales. CoRR (2018). URL http://arxiv.org/abs/1803.08240
21. Merity, S., Keskar, N.S., Socher, R.: Regularizing and optimizing LSTM language models. In: International Conference on Learning Representations (2018)
22. Merity, S., Mccann, B., Socher, R.: Revisiting Activation Regularization for Language RNNs. CoRR (2017). URL https://github.com/pytorch/examples/tree/
23. Mielke, S.J., Eisner, J.: Spell once, summon anywhere: A two-level open-vocabulary language model. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 6843–6850 (2019)
24. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, K.Q. Weinberger (eds.) Advances in Neural Information Processing Systems, pp. 3111–3119. Curran Associates, Inc. (2013)
25. Pretorius, L., Bosch, S.: Exploiting cross-linguistic similarities in Zulu and Xhosa computational morphology. In: Proceedings of the First Workshop on Language Technologies for African Languages, pp. 96–103. Association for Computational Linguistics, Athens, Greece (2009). URL https://aclanthology.org/W09-0714
26. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. Tech. rep., OpenAI (2019). URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
27. Sennrich, R., Haddow, B., Birch, A.: 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), pp. 1715–1725. Association for Computational Linguistics, Berlin, Germany (2016)

28. Shannon, C.E.: A Mathematical Theory of Communication. Bell System Technical Journal **27**(3), 379–423 (1948). DOI 10.1002/j.1538-7305.1948.tb01338.x
29. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of machine learning research **15**(1), 1929–1958 (2014)
30. Takase, S., Suzuki, J., Nagata, M.: Direct output connection for a high-rank language model. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 4599–4609. Association for Computational Linguistics, Brussels, Belgium (2018). DOI 10.18653/v1/D18-1489. URL https://aclanthology.org/D18-1489
31. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (eds.) Advances in Neural Information Processing Systems, pp. 5998–6008. Curran Associates, Inc. (2017). URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf
32. Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., Fergus, R.: Regularization of neural networks using dropconnect. In: International conference on machine learning, pp. 1058–1066 (2013)
33. Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Łukasz Kaiser, Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., Dean, J.: Google's neural machine translation system: Bridging the gap between human and machine translation. CoRR **abs/1609.08144** (2016). URL http://arxiv.org/abs/1609.08144
34. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R.R., Le, Q.V.: Xlnet: Generalized autoregressive pretraining for language understanding. In: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (eds.) Advances in Neural Information Processing Systems, vol. 32, pp. 5753–5763. Curran Associates, Inc. (2019). URL https://proceedings.neurips.cc/paper/2019/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf
35. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent Neural Network Regularization . CoRR (2015). URL https://github.com/wojzaremba/lstm