UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Final Paper 2020

Title: Comparison of LSTM architectures for modelling South African Languages

Author: Luc Hayward

Project Abbreviation: Low-LM

Supervisor(s): Jan Buys

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	0
Theoretical Analysis	*0*	*25*	0
Experiment Design and Execution	*0*	*20*	20
System Development and Implementation	*0*	*20*	10
Results, Findings and Conclusions	*10*	*20*	20
Aim Formulation and Background Work	*10*	*15*	10
Quality of Paper Writing and Presentation	*10*		10
Quality of Deliverables	*10*		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	
Total marks		**80**	

Comparison of LSTM architectures for modelling South African Languages

Luc Hayward
University of Cape Town
Cape Town, South Africa
hywluc001@myuct.ac.za

ABSTRACT

Language models form an important starting point for many natural language processing tasks. Recent work in natural language processing (NLP) has shown great advances in synthetic benchmark scores when given sufficient training data. However comparatively little work is done to improve language modelling techniques in African languages. The use of Recurrent Neural Networks (RNNs) has been shown to produce high performing language models on English and other high resource languages. This paper examines the performance of Long short term models (LSTMs) and Quasi Recurrent Neural Networks (QRNNs) and their applicability to South African low resource languages, with the aim of determining whether the increased complexity of modern techniques sufficiently improves on a more basic LSTM model. Testing across different datasets shows that the newer, more complex variants continue to significantly outperform more simplistic LSTM models. Moreover, similar relative performance in both model accuracy and training speed is seen between the LSTM and QRNN architectures as is seen in English language testing.

CCS CONCEPTS

• **Computing methodologies** → Neural networks; **Machine translation**; • **Information systems** → *Language models.*

KEYWORDS

language modelling, neural networks, low-resource languages, LSTM, QRNN

1 INTRODUCTION

Statistical language modelling is an important part of many natural language processing tasks such as information retrieval, voice recognition, machine translation, spelling correction and question answering [3, 9, 18, 21]. Whilst this has been shown extensively in languages such as English and Mandarin Chinese, resource rich languages where large datasets are available on which to test models, many languages do not have datasets as large or of equal quality. These languages are referred to as Low Resource Languages due to the relative lack of data on which to train and evaluate language models. Recent advances in language modelling have yielded significant improvements in performance, however these newer models are typically developed with resource rich languages in mind. In this paper we examine whether these improvements can also be applied effectively in low resource South African languages. Additionally this paper examines whether the change in language affects the relative performance of the models overall, and the impact of different hyperparameters for the models on their performance.

This paper examines Long Short-Term Memory models (LSTM) [12] and specifically the AWD-LSTM [17] and QRNN [2] variants which we examine in further detail under Model Choice(3.3). We hypothesise that the improved variations in LSTM models will continue to show a performance increase over the standard LSTM model on the examined South African Nguni languages, such as isiZulu and Sepedi. Specifically the improved regularisation and varying forms of dropout used by the AWD-LSTM variant to better regularise the models and reduce overfitting. For the QRNN it is hypothesised that the models will continue to train relatively faster than the LSTM model architecture without sacrificing model performance.

Long Short-Term Memory (LSTM) language models are a class of artificial neural networks and up until recently had pushed the state of the art in language modelling. While newer and more complicated LSTM variants show a clear performance improvement over the simpler basica LSTM models in a high-resource setting, it remains to be observed whether these changes continue to yield a significant improvement in the context of low-resource South African languages.

In addition to the lack of high quality data for South African languages, these languages are typologically[1] very different to the languages typically studied for language modelling. African languages can be classed as morphologically rich languages [19], languages in which grammatical relations (such as Subject, Object, etc) are indicated by changes in the words rather than the relative position of words in the sentence. Moreover african languages are agglutinative, wherein the words within the language are made up by combination of smaller morphological units or "sub-words". This leads to potentially very large vocabulary sizes where each word appears relatively few times, in spite of the "sub-words" on their own being more prevalent across the vocabulary (as they are used in the makeup of many different words).

We examined the effectiveness of the AWD-LSTM and QRNN model architectures on isiZulu and Sepedi and compared the performance against a more basic LSTM model without the newer models improvements. Further we made use of byte pair encoding (BPE) [22] to overcome the issue of extremely large word level vocabularies. BPE is a technique which allowed for arbitrary vocabulary sizes to be chosen without resulting in out of vocabulary tokens and is explained in Byte Pair Encoding (3.4). Section 2 presents background on the topics covered in this paper and related previous works. Section 3 describes the design of the experiments and the implementation of the various models and testing frameworks needed. Section 4 contains the results and a discussion thereof.

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

In section 5 we draw conclusions from the results, and finally in section 6 we propose possible paths for future work on the subject.

2 BACKGROUND AND RELATED WORK

2.1 Language modelling

A statistical language model can be defined as the probability distribution of a set of strings over a given context[4, 21], or as a model capturing the probability of a string S occurring in a given context C. Properly trained models are able to capture more complex structures such as grammatical rules based only on this statistical distribution.

2.2 Evaluating Language Models

Language models can be evaluated either extrinsically or intrinsically. Extrinsic evaluation is done through integrating the model into a real world application and evaluating whether the application improves.[14] This could be constructed as user studies in which participants may be asked to rate which generated text was "better" in order to compare the output of text generation models for instance. Intrinsic evaluation is the use of synthetic metrics to approximate the performance of a model in a real world environment, and measure relative performance against other models. The advantage of intrinsic evaluation is that they require no user feedback and can be measured independent of any specific application.

2.2.1 Perplexity. One way to measure the performance of a model intrinsically is by measuring the probabilities assigned to each token in a sequence on which the model has not been trained. In practice, perplexity is used which measures the inegative log likelihood assigned to the test set, normalised for the number of words in the test set. [14] As perplexity is an *inverse* probability, *minimising* perplexity scores is equivalent to better model performance over the test set. Although this is not guaranteed to translate to better performance in real world applications, it is much faster to calculate perplexity and is an appropriate analogue when comparing models initially.

2.2.2 Cross Entropy. Entropy can be thought of as the amount of information encoded by some x_i, alternatively lower entropy is indicative of an event or value with a high probability of occurring whilst a high entropy value implies the event occurs less frequently. Measuring the entropy of a language requires measuring infinitely long sequences, however the Shannon-McMillan-Breiman theorem states an acceptable estimate of the entropy of a language L can be found for a sequence of length n which is "long enough". Following this, the cross entropy of a model m on some distribution p can be calculated as [14]:

$$H(p, m) = -\lim_{n \to \inf} \frac{1}{n} \log m(w_1 w_2 ... w_n)$$

Cross entropy can be thought of as a measure of the overlap in entropy between two sets. In the case of language modelling, cross entropy measures how well the entropy of the test set is captured by the language model being examined. A lower cross entropy value represents the model which best models the test sets distribution. [21]

2.2.3 Bits Per Character. The bits per character or BPC is another measure of the cross entropy of a language model. BPC attempts to model the performance of models across vocabulary sizes, although as with other measures direct comparisons across datasets is not usually possible. The BPC is derived from the cross entropy of the model, converted to base 2 if not originally calculated as such, divided by the average number of characters per token for models which are not character based. This allows for models to be compared relative to one another in order to determine the optimal vocabulary size.

2.3 Model Architectures

2.3.1 N-Grams. N-gram models are one of the simplest language models, and are based on the Markov assumption that rather than calculate all possible sequences of words, a model can approximate the distribution of a language using a shorter sequence length N to provide context [14]. This is done by basing the probability of a word S in a sequence, on its frequency relative to the frequency of all other sequences of the same length in the training corpus. More formally:

$$P(S_i) = P(S_i | S_{i-1} ... S_{i-1-n})$$

Where n is the number of words (or values in the sequence) used to provide context.
N-gram model complexity grows exponentially as more words are considered in a sequence (increasing N). Additionally as sequence length grows, the size of the training corpus must grow in order to ensure there are sufficient examples of sequences to create an accurate model. The simplicity of n-gram models leads to trouble capturing long term dependencies where there are large gaps between the relevant parts of a sequence. This is due to the exponential increase in the number of possible n-grams (and thus model size) as you increase the context window [6, 24].

2.3.2 Recurrent Neural Networks. In the buildup to more advanced neural network based language models, n-gram models can be replaced by a simple feed forward neural network (FFNN) [1, 13]. By concatenating the previous $n - 1$ word vectors as input to the network, it is possible to train the network to predict (or classify) the next word in the sequence. This is in effect a "neural n-gram". This technique introduced its own limitations, namely that the dimensionality of the input vector is directly related to the vocabulary size and the context length. [11]

A more appropriate network which can be used in place of FFNNs for sequence modelling is the Recurrent Neural Network (RNN). A benefit of RNN models over the described FFNN is that at each time step only one token in the sequence is fed as input, preventing the need for very large input vectors as context length is increased. RNNs are well suited to the sequential process of language modelling thanks to the use of a shared hidden state or persistent model across time steps [12]. Between each timestep the RNN model passes the output of the hidden layer as input to the hidden layer in the next timestep, allowing previous inputs to influence future timesteps outputs. (Figure 1)

This intuitively allows for modelling of sequential data as the output at each time step depends not only on the current input, but all those inputs that come before it. Critically, RNN models are not

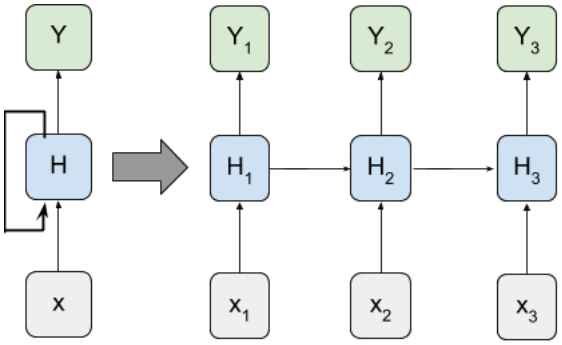


Figure 1: RNNs can be "unrolled" better showing the flow of information between timesteps

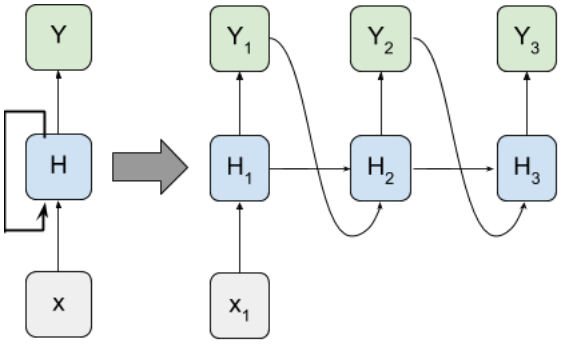


Figure 2: A simple RNN network can be "unrolled" as a sequence of RNN cells, the output of each cell being fed as input to the next

limited to a fixed input or output sequence length allowing them to adapt to different sequence-to-sequence tasks.

Formally an RNN can be described as follows: [10]

$$\begin{aligned} a^t &= b + Wh^{t-1} + Ux^t \\ h^t &= \tanh(a^t) \\ o^t &= c + Vh^t \\ \hat{y}^t &= softmax(o^t) \end{aligned} \tag{1}$$

Where {(t)} is the timestep, {x} is the input. {b} and {c} are bias vectors whilst {U}, {V}, and {C} are the weight matrices for the input-to-hidden, hidden-to-input and hidden-to-hidden connections respectively. {o} is the unnormalized log probabilities of each value of the discrete variable while {y} is the normalized probabilities over the output.

When applied to language modelling, an additional change is often made where the output from the network is fed directly as input into the network in the following timestep in cases where the model is intended to generate sequences of tokens itself.(Figure 2) [25]

As each hidden layer output contributes to the result of the hidden layer at the next time step, during training this needs to be accounted for in the loss function. The backpropagation through time (BPTT) algorithm allows for each hidden layer to contribute to the loss at a fixed number of previous timesteps. This repeated multiplication gradually pushes the gradients towards zero as sequence lengths increase. [15, 27] This is known as the "Vanishing gradient problem" in which during training the gradients tend towards zero, preventing the model from learning to encode long context sequences. Variations such as LSTM models aim to increase the model's ability to model long term dependencies and thus improve performance on sequence modelling tasks.

2.3.3 Long Short-Term Memory. Long Short-Term Memory (LSTM) models are a popular variation on the RNN architecture allowing for longer term dependencies to be modelled effectively. An LSTM network expands on the traditional RNN architecture by introducing the "context" or "memory" vector which is passed between timesteps in addition to the usual hidden layer vector. Put simply the LSTM context vector encodes long term dependencies by allowing the model to explicitly "decide" to add or remove information from the context vector at each time step.

A standard LSTM cell, the computational unit for a single step, with input and state size m and n respectively can be described more formally as in Melis et al. : [16]

$$LSTM : R^m \times R^n \times R^n \rightarrow R^n \times R^n$$

$$LSTM(x, c_{prev}, h_{prev)} = (c, h)$$

The updated state c and the output h can then be computed as follows:

$$\begin{aligned} f &= \sigma(W^{fx}x + W^{fh}h_{prev} + b^f) \\ i &= \sigma(W^{ix}x + W^{ih}h_{prev} + b^i \\ j &= \tanh(W^{jx}x + W^{jh}h_{prev} + b^j \\ o &= \sigma(W^{ox}x + W^{oh}h_{prev} + b^o) \\ c &= f \odot c_{prev} + i \odot j \\ h &= o \odot \tanh(c) \end{aligned} \tag{2}$$

where σ is the logistic sigmoid function, $\odot$ is the elementwise product, W^{**} and b^* are weight matrices and biases.

A major difficulty for LSTMs (and all RNN based models) is the inherently sequential nature of the network. This necessitates that the model be trained on sequences of input, making parallel training and prediction more difficult as each step in the training is impacted by each previous step in the sequence. This becomes particularly apparent in large models as sequence length grows in order to capture increasingly longer term dependencies, due to memory constraints. Additionally, without proper regularisation's LSTMs often suffer from overfitting the training data resulting in poor performance on unseen datasets. [17]

2.3.4 Regularisation in RNNs. Regularisation describes a set of techniques used to improve generalisation performance of neural networks and prevent overfitting. Attempts to apply standard techniques such as *dropout* [23] are not successful when applied between timesteps on the recurrent connection as it inhibits the models ability to retain long term dependencies. [17, 28] Merity et al. introduced the use of a weight-dropped LSTM using DropConnect[26] on the hidden-to-hidden weights. 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 [17].

Merity et al.[17] examined a range of regularisation techniques, of which variational dropout[8] had the greatest impact on model performance based off model ablation testing. Variational dropout generates a dropout mask once which is then used over the entire forward and backward pass, rather than resampling at every timestep. For the AWD-LSTM model, Merity et. al. proposed a combination of DropConnect for the hidden-to-hidden transitions within the LSTM and variational dropout over the inputs and outputs. Drop connect has a similar result as variational dropout (where the is applied to the recurrent connections), although here the dropout is applied to the recurrent weights. This helps to prevent the overfitting of the recurrent connections.

2.3.5 AWD-LSTM. The AWD-LSTM is a variant on the standard LSTM architecture which incorporates the improvements to the regularisation techniques proposed by Merity et. al. These include the *Variational Dropout* and *Drop Connect* techniques above as well as *Variable length backpropagation sequences*, *Embedding Dropout*, *Weight Tying* and *Activation and Temporal Activation Regularization*.

Variable length backpropagation sequences involves slightly changing the sequence length for the forward and backward pass such that the starting point for the BPTT window does not always fall on the same tokens, leaving part of the dataset (1/sequence_length specifically) unused. By changing the sequence length slightly, this ensures that, given enough epochs of training, all elements of the training set will be used. Embedding dropout is equivalent to performing dropout on the embedding matrix at the token level, and can be thought of as removing all instances of a given token T from the sample. Activation and Temporal Activation Regularization refer to L2 regularization and L2 regularization decay respectively. L2 regularization is often used to control the network weights to reduce overfitting by controlling the norm of the model.

2.3.6 Quasi-Recurrent Neural Networks. Inspired by the RNN models, Quasi-Recurrent Neural Networks (QRNNs) [2] allowed for parts of the typical RNN architecture to be computed in parallel. It was found that the models were capable of state of the art results on the smaller PTB and WT2 datasets while requiring shorter training times, seeing as much as as 16x speedup over conventional RNNs (longer sequence lengths led to greater differences in speed). Similar performance on the larger word-level datasets or in character level models was not found.

2.3.7 Transformer. Transformers are a relatively new form of language model which rely solely on attention mechanisms, removing the need for recurrence in the networks [25]. This results in a more parallelisable model allowing for far faster training and consequently enabling much larger models. Furthermore the parallel nature of the model allows for much longer sequence lengths, this coupled with the attention mechanism allows for dependencies to be modelled over longer contexts than would be possible in an RNN/LSTM network. In addition to modelling longer term dependencies, transformer self attention layers are faster to compute than recurrent layers when the sequence length is smaller than the embedding dimensionality.

Due to their ability to model long range dependencies, Radford et al. [20] demonstrated that a transformer model trained on a sufficiently large dataset was capable of encoding information required to perform a variety of language modelling tasks such as question answering and machine translation. Despite the performance achieved, training these models is extremely computationally intensive. Specifically the largest model produced, GPT-3, is an orders of magnitude larger than the previous GPT models produced by OpenAI.

3 DESIGN AND IMPLEMENTATION

3.1 Training Data

The Sotho-Tswana and Nguni language families are two of the largest language families in South Africa, as such Sepedi and isiZulu respectively were chosen as representative languages due to having the largest dataset of similar size of the potential languages. Standard practice was followed for research in language modelling, the datasets were split into training, validation and test sets with the BPE model trained only on the training set. The datasets were collected from three separate sources. This allowed for the models to be tested across different datasets. A comparison of the various dataset sizes is available in table **??**.

NCHLT: The South African Centre for Digital Language Resources (SADiLaR) [2] provides monolingual corpora for all 11 of South Africa's official languages. Corpora for Sesotho, Sepedi, isiZulu, Siswati, Setswana, isiNdebele, tshiVenda and isiXhosa were collected for the 2014 National Centre for Human Language Technology (NCHLT) Text project [5]. A significant proportion of the texts are scraped from governmental websites. The corpora range in size from 1 to 3 million tokens. The Sepedi and isiZulu corpora were used from this dataset. NCHLT is the largest of the three datasets, with the isiZulu corpus having approximately ten thousand more sentences but forty five thousand fewer total words than the Sepedi corpus contained.

Newstools Isolezwe Corpus: News articles from the isiZulu Isolezwe newspaper have been scraped and consolidated for the newstools initiative[3]. The process has been automated and the repository of news articles is updated on a continuous basis as news articles are published. At the time of writing, we estimate that this corpus contains under 1.2M words of acceptable quality. However this provided only a Sepedi dataset and not isiZulu. The Isolezwe dataset is comparable in the number of sentences to the NCHLT-Sepedi corpus, though notably still has a similar total word count as the NCHLT-isiZulu corpus.

Autshumato: The Autshumato[4] dataset is aproximately one third the size with respect to the total sentence counts, compared to the other datasets chosen. Of note, the total word count for isiZulu is approximately one quarter that of the other two datasets, whilst the Sepedi corpus contains many more words and is only one third the NCHLT-Sepedi corpus size in this regard. The data is drawn for the South African Government domain and is presented as parallel

[2] Datasets are available at www.sadilar.org
[3] Available on Github at https://github.com/newstools
[4] Available at https://sourceforge.net/projects/autshumato/files/

corpora in a number of languages against English. This paper used only the isiZulu and Sepedi data, English was also not used.

3.2 Data Processing

After downloading the data for the project from the different repositories, each dataset goes through a data preprocessing step. In this preprocessing stage a number of steps are taken to prepare the data and normalise any inconsistencies. As most of the datasets originate from articles scraped from the web, or scanned from government documents, many instances of English, HTML and Javascript lines, and other repetitive or erroneous data are mixed in. As these would not naturally be found across general language they were removed prior to splitting the datasets. After downloading the data, a number of lines was stripped from the start of each dataset to remove any header info as well as removing the constitution from the Autshumato datasets due to the sentences being poorly formatted and out of order. The data was then cleaned by normalising punctuation (such as quotations or ellipses), removing numberings (such as "1.3.14" and (vii)) and trimming any whitespace or non-alphabetic characters from the start of each sentence. Finally sentences which were either too short (representing sections and headings) or significantly too repetitive were removed. These parameters were chosen experimentally to maximise both quality and quantity of the remaining data and are different for each data source.

Each dataset was then split into a training, validation and test set using an 80, 10, 10 split. The splits were done using sequential blocks for each dataset, rather than a random permutation of sentences. This was done so as to preserve the order or sentences and allow for different batch sizes and sequence lengths to be used by the models without changing the data processing stage. By keeping the splits consistent across the different model runs no inconsistencies due to changes in the training, validation or test data used that could affect the models.

3.3 Model Choice

The initial choice of model centered around the AWD-LSTM model proposed by Merity et. al. [17]. This was chosen in lieu of a more recent model as it represents a foundational model upon which more recent LSTM variants have improved. As additional models against which the relative performance could be compared, it was decided to explore the QRNN implementation from Merity et. al. as well as a stripped down, basic LSTM model. The simpler LSTM, without any of the improvements of the AWD-LSTM variant, was created as baseline model. This model, the "Basic LSTM", would serve as a reference point against which to compare the two more complex models and determine whether the added complexity (and thus training time) was offset by improved performance. These three models were chosen in order to test whether similar relative results would be found on our datasets as was found by Merity et. al. on the PTB and Wikitext datasets when comparing the AWD-LSTM and QRNN based models. Namely it is hypothesised that the QRNN model will train faster whilst achieving equivalent or better perplexity scores than the AWD-LSTM, with both models significantly outperforming the basic LSTM.

3.4 Byte Pair Encoding

Byte Pair Encoding (BPE) [7] was originally designed as a compression technique, where the most frequent pair of bytes are replaced with new unused byte and the substitution stored for later reference. Sennrich et. al. [22] showed a variation of this algorithm could be applied to word segmentation. In this method, characters rather than bytes are merged together, one pair at a time, after starting with an initial character level vocabulary (as well as a special end of word symbol). For example, if the character pair ("A", "B") was the most frequent pair this would be combined as a new symbol "AB" and the merge would be recorded. The resulting merges represent "subwords" of the vocabulary. Intuitively this can be though of in a similar manner as making the word "colder" from the tokens "cold" and "er". A BPE model has only one parameter, the final size of the vocabulary. As each BPE model starts with a character level vocabulary, the final vocabulary will be capable of encoding any word in the original corpus as well as any unseen words. In such a way BPE allows for the flexibility of an open vocabulary as with character level models, as well as the efficiencies of word level models whereby words can represented with fewer tokens.

For each model and dataset, a BPE model wast trained using a vocabulary size of 5000 tokens, except in certain specific experiments intended to determine a good vocabulary size for the models. The BPE model was trained using only the training set for each dataset. This was done to prevent the encoder from learning pairs that may have been common in the validation or test set but did not appear as often or at all in the training set. This would have resulted in poorer performance by the models.

3.5 Overview of Experiment

The LSTM models were implemented in Pytorch and make use of the AWD-LSTM framework [17], whilst the BPE preprocessing uses the HuggingFace Library[5]. Models were evaluated intrinsically by comparing the perplexity and BPC (Bits per character) scores between the three models across the different datasets. In order to train the models efficiently, Google Colab [6] and later the UCT CHPC [7] were used for GPU access. The use of GPU training was essential in order to allow for enough models to be in a timely manner.

3.6 Model Evaluation

Models were to be evaluated intrinsically by comparing the BPC and the perplexity of the models against each other. Specifically, by comparing the performance of a basic LSTM network to the more advanced QRNN and AWD-LSTM networks, it can be determined whether these more complex networks would continue to yield improved performance despite the low resource nature of the datasets on which the networks are trained. In the case of results for which the vocabulary sizes are different, the BPC scores were used to approximate the performance across the models. This was done by taking the bits per token value and normalising it by the average token length for the dataset. The different models could thus be

[5] Available at https://github.com/huggingface/tokenizers
[6] Accessible at https://colab.research.google.com/
[7] Available at https://www.chpc.ac.za/

compared against each other on the same dataset, however comparisons across languages or datasets are not meaningful beyond comparing the relative performance of models on their respective datasets. For instance model *A* may be the better model for a given dataset than model *B* whilst model *B* was better than model *A* on an alternate dataset.

As the models examined are based largely or entirely off of the frameworks made available by previous studies, the model implementations were first trained and tested on the Penn Treebank (PTB) datasets and compared against the expected performance as reported in the respective papers. This ensures that there were no errors in the implementations that could affect model performance as some changes to the code had to be made in order to add the BPE encoders and fix some errors in the regularisation parameters. Additionally the models were trained on systems using an unsupported version of CUDA which may have impacted the performance. However it was determined that neither of these factors affected the models with both the AWD-LSTM and QRNN performing as reported by Merity et. al.

4 RESULTS AND DISCUSSION

In this section, the results of the experiments on the different models and their parameters are presented across the datasets and languages. We discuss the implications of these results, possible explanations for the outcomes and limitations of the experiments. Detailed tables of results are available in the supplementary information C. We examine each section and language largely in isolation. This is due to the fact that comparing the results of language models across datasets does not lead to meaningful comparisons. Rather each dataset is considered in isolation and we will attempt to draw some general conclusions in the final section regarding any generalities in relative performance across the datasets.

4.1 NCHLT

The NCHLT dataset represents the largest portion of the experiments tested due to its large size and varied, quality corpus of data to work from. As such the testing and fine tuning of the models parameters for each of the models tested on the other datasets was refined initially on the NCHLT dataset. For both the QRNN and AWD-LSTM models, the starting point was the parameters chosen by Merity et. al. for their word level wikitext2 dataset. This was chosen as it was the most similarly sized dataset to the NCHLT isiZulu corpus. These results were effective but found to be improvable with larger models and longer training times. Although each models results were stored, due to an undiscovered data error the test results for the AWD-LSTM variant models for both NCHLT languages were not useable in the final results. The conclusions drawn below are thus based solely off the validation data and cannot be used as accurate or fair representations of the models performance on unseen data when trained on this dataset.

4.1.1 isiZulu. The isiZulu corpus was chosen as the corpus for the majority of testing when looking for the best model parameters as it is present in both of the other two datasets used.

QRNN: The best model architecture overall was the QRNN model, with the best models performing at least 0.08 bpc points (approximately 5% higher than the lowest bpc values) better than either the AWD-LSTM or basic LSTM model variants.A number of variations on the initial parameters were run in order to attempt to find the best validation results. These included longer training times, changing the embedding size, changing the vocabulary size, changing the number of hidden layers and changing the batch size (see Table 1). Ultimately none of these variations produced significantly better results than simply increasing the total size of the model by either increasing the number of hidden layers or increasing the input embedding size. Additionally, the tests involving the changing of the vocabulary size were shown to not have a significant effect on the overall model performance on the validation set. As such going forward the QRNN model used the parameters of the model labelled "wt2InspiredQRNNLongTrain", with a typical vocabulary size of 5000, although the extra training length was omitted from subsequent testing due to time constraints and a lack of evidence that it significantly improved model performance based off the validation set.

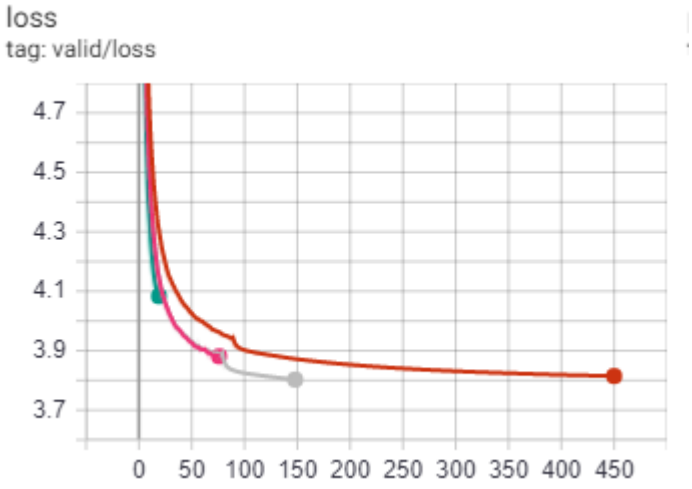


Figure 3: Y-axis: Validation loss, X-Axis: epoch. A comparison of the AWD-LSTM model variants' performance on the validation set as training continues. The orange line represents the base AWD-LSTM model without any adjustments to the intial parameters. The green, pink, grey line is the same model, although the grey was able to train for longer.

AWD-LSTM: A similar process as for the QRNN was followed when optimising the AWD-LSTM model parameters. Figure 3 shows the progression of validation score as the models train. The models shown are the base AWD-LSTM model and a larger model (double the embedding size at 800). Unlike the QRNN models where increasing the model size did not drastically affect trianing speed, here the AWD-LSTM is greatly impacted by the size of the model. Although the larger model appears to train in far fewer epochs, it should be noted that each epoch takes significantly longer. Though both models were trained in approximately 12 hours, the larger model took approximately 1.5 times as long per epoch. It was after training this model that we were able to gain access to the UCT CHPC which allowed for models to be trained far faster and more

Val Order	descriptive_name	Test_BPC	emsize	nhid	nlayers	batch_size	bptt	dropouth	wdrop	nonmono	vocab_size	Val_BPC
1	wt2InspiredQRNNLongTrain	1.323	**800**	1550	4	40	70	**0.2**	**0.1**	8	10000	1.169039
2	testVocab7QRNNmk2	1.323	**800**	1550	4	40	70	**0.2**	**0.1**	8	7500	1.169542
3	wt2InspiredStuBiggerP100mk2	1.324	**800**	1550	4	40	70	**0.2**	**0.1**	8	5000	1.169857
4	testVocab2QRNNmk2	1.326	**800**	1550	4	40	70	**0.2**	**0.1**	8	2500	1.17255
5	testVocab10QRNNmk2	1.33	**800**	1550	4	40	70	**0.2**	**0.1**	8	10000	1.177129
6	wt2InspiredStuBiggerP100mk3SmallerEmbMoreLayers	1.339	**400**	1550	**6**	40	70	**0.2**	0.1	8	5000	1.183812
8	wt2InspiredQRNNBiggerBatch	1.339	400	**1550**	4	**80**	70	0.2	0.1	5	5000	1.195434
13	wt2InspiredQRNNBiggerBatch	1.4	400	1550	4	**40**	**70**	**0.4**	**0.2**	5	5000	1.245021
14	wt2InspiredQRNN_LongerBPTT	**1.404**	400	**1200**	**4**	40	**120**	0.2	0.1	5	5000	1.250593
19	wt2InspiredQRNN_NoRegularisation	1.548	400	1550	4	40	70	0	0	5	5000	1.392733
7	wt2InspiredQRNN	10	400	1550	4	40	70	0.2	0.1	5	5000	1.186465

Table 1: The different parameter tuning tests run on the QRNN model for NCHLT-isiZulu. Values in bold are changed from the original QRNN parameters. Parameters which are consistent across all models are ommitted for brevity. The initial wt2Inspired QRNN model was not tested beyond the validation set but is included for completeness. The NoRegularisation model does not have any regularisation.

efficiently. As such in spite of the far longer training times it was decided to continue the experiments using the larger model variant.

Basic LSTM The basic LSTM is a stripped down version of the AWD-LSTM model. All additional regularisation techniques were disabled leaving only the input dropout enabled. This serves to provide an example of an older, more simple LSTM model. The relative performance of this basic LSTM model would allow it to be determined whether the added training time and complexity of the two more complex models was effective. Our hypthesis that the model would perform worse than the two more complicated models was upheld during initial testing on this dataset. In addition to the basic LSTM with input dropout, we also exmined a basic LSTM with only variational dropout as well as with only embedding dropout. It was found that the embedding and input dropout variants performed similarly with only very minor changes in bpc results on both the validation and test sets. This was expected as input and embedding dropout both can be thought of as having similar effects, reducing or removing a tokens impact on the model for a given sequence. However the Variational only dropout showed slightly better results of approximately 0.03 bpc on both validation and test sets. This is likely due to variational dropout affecting many more weights throughout the model than the other two dropout methods at each step regardless of which words appear in the sequence.

4.1.2 Sepedi. Having completed the parameter tuning and deciding upon the final models to test across the other datasets using the results gathered from the isiZulu testing process. The relative results of the models on the validation sets were the same as on the isiZulu corpus, supporting the hypothesis that due to the similar nature of isiZulu and Sepedi (both agglutinative African languages), models would likely perform similarly when trained for either language task. In this instance the difference in performance between the QRNN, Variational dropout-only LSTM and input dropout LSTM were also very close to the results from the isiZulu corpus. Again this is likely due to the language similarities.

With respect to the basic Input Drop LSTM model, due to the increased speed of the CHPC it was possible to run an additional variant using a deeper LSTM with 3 hidden layers (up from one on the basic model). However unlike the similarly sized AWD-LSTM or QRNN models, this deeper basic LSTM trained erratically, rapidly fitting the training data before stabilising at similar training scores as the other basic LSTM models (see Figure 4). However the model

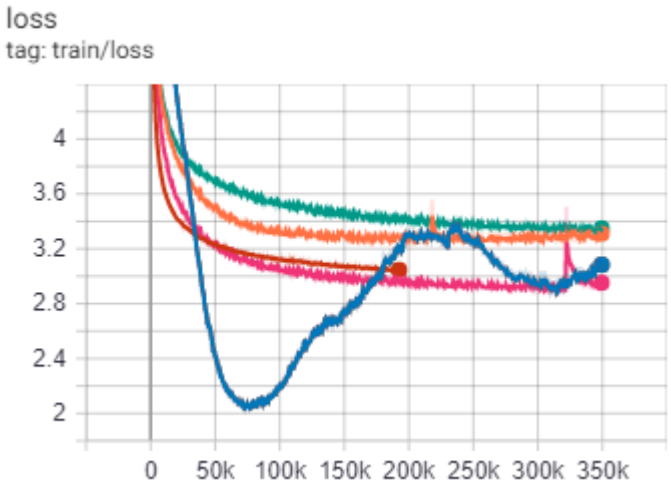


Figure 4: Y-axis: Training loss, X-Axis: epoch. Graph displaying the erratic training loss of the deep basic LSTM model (blue) compared to the other LSTM models.

ultimately was unable to accurately model the validation data and quickly diverged (see Figure 5)

4.2 Isolezwe

The Isolezwe dataset contained only an isiZulu corpus, no equivalent Sepedi corpus was available. Isolezwe is roughly equivalent to NCHLT in size and it was hypothesised that the results of the different models tested would be equivalent to those seen on the NCHLT isiZulu corpus. This hypothesis was upheld with respect to the QRNN readily outperforming the other models tested. However the AWD-LSTM model, though performing as expected relative to the other models on the validation set, was unable to model the validation set and was a full 1.68 bpc behind the QRNN and 0.8 bpc worse than the next worst performing model (the "deep" basic (input dropout) LSTM). This is a significant difference in performance as bpc is on a log scale. It is not clear what caused the model to perform this much worse than would be expected.

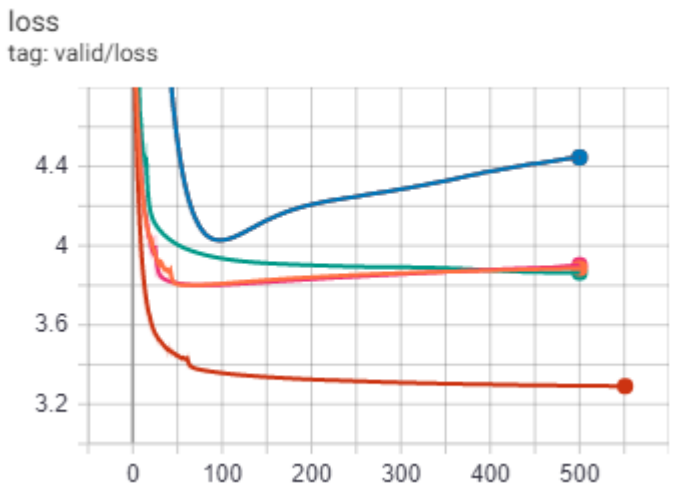


Figure 5: Y-axis: Validation loss, X-Axis: epoch. Graph displaying the growing inability of the deep basic LSTM to model the validation dataset (blue) compared to the other LSTM models (green: basic input dropout, orange: variational dropout only, red: AWD-LSTM model type).

4.3 Autshumato

The Authsumato dataset, being noticeably smaller than any of the other datasets regardless of language, represents a much more difficult modelling exercise. We hypothesised that a smaller AWD-LSTM model would outperform the other model variants tested due to the lack of training data available for the larger QRNN model. However the QRNN outperformed all other models on the isiZulu corpus and was within 0.008 bpc points of the AWD-LSTM model variant on the larger sepedi corpus. Similarly the basic LSTM model variants once again fell approximately 0.15 bpc behind the other two models as in the other datasets (An interesting similarity that the difference in bpc across the datasets remains steady, but it should not be taken as a sign of any greater meaning beyond coincidence due to similarly sized and constructed datasets. Direct comparisons across datasets would not yield meaningful results). As with the QRNN model, the isiZulu dataset though far smaller than the Sepedi dataset resulted in similar relative performance among the basic LSTM models as the other datasets, with the variational only dropout model again outperforming the basic Input Dropout and deep basic input dropout models. However the Sepedi corpus tests resulted in the variational dropout underperforming all other basic LSTM models tested. These results were unexpected as the significantly smaller isiZulu corpus was expected to be more likely to lead to different results than the other datasets given that it was only one quarter the size due to large sections of badly constructed test scans.

4.4 General Discussion

Overall the QRNN model architecture was the top performing model in across all datasets and languages tested, with the exception of the autshumato sepedi dataset. On the larger NCHLT and Isolezwe datasets, the QRNN model outperformed the other models by at least 0.15bpc which represents a 10-15% increase in bpc for the runner up model relative to the QRNN results. However for the both autshumato datasets, the QRNN produced a very similar result to the large AWD-LSTM model, with the AWD-LSTM in fact producing a lower bpc (albeit by 0.008 points) than the QRNN in the autshumato-Sepedi dataset. It is possible that the significantly reduced size of the Autshumato dataset did not allow for the QRNN to perform as well as it appears able to in the larger datasets relative to the AWD-LSTM models.

In general the relative performance of the various models within the datasets on the validation set was largely maintained. This suggests that the chosen splits still allowed for an accurate representation of the dataset in each split. With no one split being decidedly different to each other. This is to be expected as each dataset was drawn from similar sources, and specifically the Validation and Test sets were split from the last 20% of each corpus.

For all models across the datasets, although longer training times were needed to produce the best possible results, the vast majority of the models training improvements were completed within the first 3 hours of the 12 hour training cycles used. This suggests that the models could continue to improve if there was far more training data available. This is expected as it is well known in other language modelling tasks that this is a relatively small amount of training data to be working with. Secondly this shows that in future works on similar datasets, likely far more parameter tuning could be done by stopping the training early as the relative performance of the models typically did not change as training went on longer (see Figure 6).

The hypothesis that the two more complex models would outperform the basic LSTM held in all test cases. This was not surprising as though it is difficult to get large amounts of training data for these languages, the training datasets we were able to collect were still relatively similar in size to the English sets used by Merity et. al., namely wikitext2. The only outlier contrary to this hpothesis was that of the Isolezwe-isiZulu tests, in which the larger AWD-LSTM model was the worst performing of the three model types on the test set. This was doubly unusual as for the other datasets, the relative ordering of the three models was largely maintained (or the models performance was with less than 0.01 bpc points) between the validation and test set results. However in this case though the relative validation results were consistent with the other datasets, the AWD-LSTM underperformed compared to the basic LSTM models.

The limitations of these results are typical of language modelling investigations and machine learning as a whole. More data and more time tune hyperparameters would always be useful in improving the breadth and reliability of results. The unexpected outliers in the results of the isolezwe and autshumato datasets could be more accurately examined by running multiple similar models to ensure that the results could be consistently replicated and possible causes examined. Further large datasets may contain other erroneous or malformed sections as was the case in the Autshumato datasets, which may have been missed due to the size of the datasets. Having access to more reliable datasets structured as parallel corpus' (as with NCHLT) would further improve the examination of the models across different languages.

5 CONCLUSIONS

The conducted experiments demonstrated that the improvements in regularization techniques and model architectures for RNNs over the standard LSTM model continue to improve model performance significantly when applied to African languages such as isiZulu and Sepedi. Furthermore it was shown that the relative improvements of QRNN models over the AWD-LSTM model variants seen in English remain in African Language modelling tasks. Namely the QRNN continues to train faster and produce equal or better results than other models tested. These results are consistent with those reported by previous works such as in Merity et. al. These relative performance improvements are consistent across a range of dataset sizes in both languages. This suggests that further improvements in RNN based language modelling would likely be directly applicable in low resource African languages going forward, without the need to repeat the long testing process needed to optimise these new models.

Additionally we show that the use of BPE to allow for open vocabulary language modelling is an effective method to account for the large word level vocabulary sizes of agglutinative African Languages.

6 FUTURE WORK

The models examined in this paper, though once state of the art, have since been iterated and improved upon further. Further work could involve examining these more complex models to determine whether there is sufficient training data to make use of their increased complexity. Alternatively, this paper examined only intrinsic measures of model performance. Future works may choose to explore a more user-centric study in which the trained models are used to generate sequences of text which could be evaluated for accuracy, readability and "humanness" metrics.

ACKNOWLEDGMENTS

The author thanks Dr. Jan Buys for his patient guidance and supervision during this research. This work is based on the research supported in part by the National Research Foundation of South Africa as well as the University of Cape Town.

REFERENCES

[1] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, Christian Jauvin, Jauvinc@iro Umontreal Ca, Jaz Kandola, Thomas Hofmann, Tomaso Poggio, and John Shawe-Taylor. 2003. *A Neural Probabilistic Language Model.* Technical Report. 1137–1155 pages.

[2] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2017. QUASI-RECURRENT NEURAL NETWORKS. In *International Conference on Learning Representations.* arXiv:1611.01576v2

[3] 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* (Kolkata, India) *(FIRE '16).* Association for Computing Machinery, New York, NY, USA, 16–23. https://doi.org/10.1145/3015157.3015160

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

[5] 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).* European Language Resources Association (ELRA), Reykjavik, Iceland, 3698–3703. http://www.lrec-conf.org/proceedings/lrec2014/pdf/1151_Paper.pdf

[6] Gernot A. Fink. 2014. Integrated Search Methods. In *Markov Models for Pattern Recognition: From Theory to Applications* (2nd ed.), Sameer Singh and Sing Bing Kang (Eds.). Springer, Chapter 12, 216. https://doi.org/10.10007/978-1-4471-6308-4

[7] Philip Gage. 1994. A New Algorithm for Data Compression. *C Users J.* 12, 2 (Feb. 1994), 23–38.

[8] 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* (Barcelona, Spain) *(NIPS'16).* Curran Associates Inc., Red Hook, NY, USA, 1027–1035.

[9] Ankur Gandhe, Florian Metze, and Ian Lane. 2014. Neural Network Language Models for low resource languages. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH.* Carnegie Mellon University, International Speech and Communication Association, 2615–2619.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. [n.d.]. Deep Learning. https://www.deeplearningbook.org/

[11] Kenneth Heafield. [n.d.]. N-Gram Language Modelling including Feed-Forward NNs. https://ufal.mff.cuni.cz/mtm16/files/08-n-gram-language-modeling-including-feed-forward-kenneth-heafield.pdf

[12] Sepp Hochreiter and Jürgen Schimdhuber. 1997. Long Short-Term Memory. *Nerual Computation* 9, 8 (1997), 1735–1780. https://doi.org/doi.org/10.1162/neco.1997.9.8.1735

[13] Yinghui Huang, Abhinav Sethy, and Bhuvana Ramabhadran. 2017. Fast Neural Network Language Model Lookups at N-Gram Speeds. In *Interspeech.* https://doi.org/10.21437/Interspeech.2017-564

[14] Dan Jurafsky and James H. Martin. 2015. Language Modeling with N-Grams. In *Speech and Language processing* (2nd ed.). Chapter 4.

[15] Dan Jurafsky and Jame H. Martin. 2019. *Speech and Language Processing* (3rd ed.). https://web.stanford.edu/{~}jurafsky/slp3/

[16] Gábor Melis, Tomáš Kočiský, and Phil Blunsom. 2020. MOGRIFIER LSTM. In *ICLR.* arXiv:1909.01792v2

[17] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and Optimizing LSTM Language Models. In *International Conference on Learning Representations.*

[18] 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.* 1–10.

[19] 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.* Association for Computational Linguistics, Athens, Greece, 96–103. https://www.aclweb.org/anthology/W09-0714

[20] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. [n.d.]. *Language Models are Unsupervised Multitask Learners.* Technical Report. https://d4mucfpksywv.cloudfront.net/better-language-models/language{_}models{_}are{_}unsupervised{_}multitask{_}learners.pdf

[21] Alessandro Scarcella. 2018. *Recurrent neural network language models in the context of under-resourced South African languages.* Ph.D. Dissertation. University of Cape Town.

[22] 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).* Association for Computational Linguistics, Berlin, Germany, 1715–1725.

[23] 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 (2014), 1929–1958.

[24] G Tur and R De Mori. 2011. *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech.* Wiley. https://books.google.co.za/books?id=RDLyT2FythgC

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30,* I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett (Eds.). Curran Associates, Inc., 5998–6008. arXiv:1706.03762v5 http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[26] 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.* 1058–1066.

[27] P. J. Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proc. IEEE* 78, 10 (1990), 1550–1560.

[28] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2015. RECURRENT NEURAL NETWORK REGULARIZATION. (2015). arXiv:1409.2329v5 https://github.com/wojzaremba/lstm.

Appendices

A DATASET SIZES

		Sentences			Words		
		Total	Training	Valid/Test	Total	Training	Valid/Test
Autshumato	isiZulu	27.5	22.0	2.8	330.8	264.6	33.1
	Sepedi	33.9	27.1	3.4	684.2	547.4	68.4
Isolezwe	isiZulu	76.9	61.5	7.7	1175.2	940.2	117.5
NCHLT	isiZulu	96.8	77.5	9.7	1223.2	978.6	122.3
	Sepedi	79.0	63.2	7.9	1696.6	1357.3	169.7

Table 2: A table showing the sizes (in thousands) of the used datasets. The sizes of each of the datasets are as found by manual inspection, with the training and Validation/test sets estimates in an 80/10/10 split.

B ADDITIONAL GRAPHS

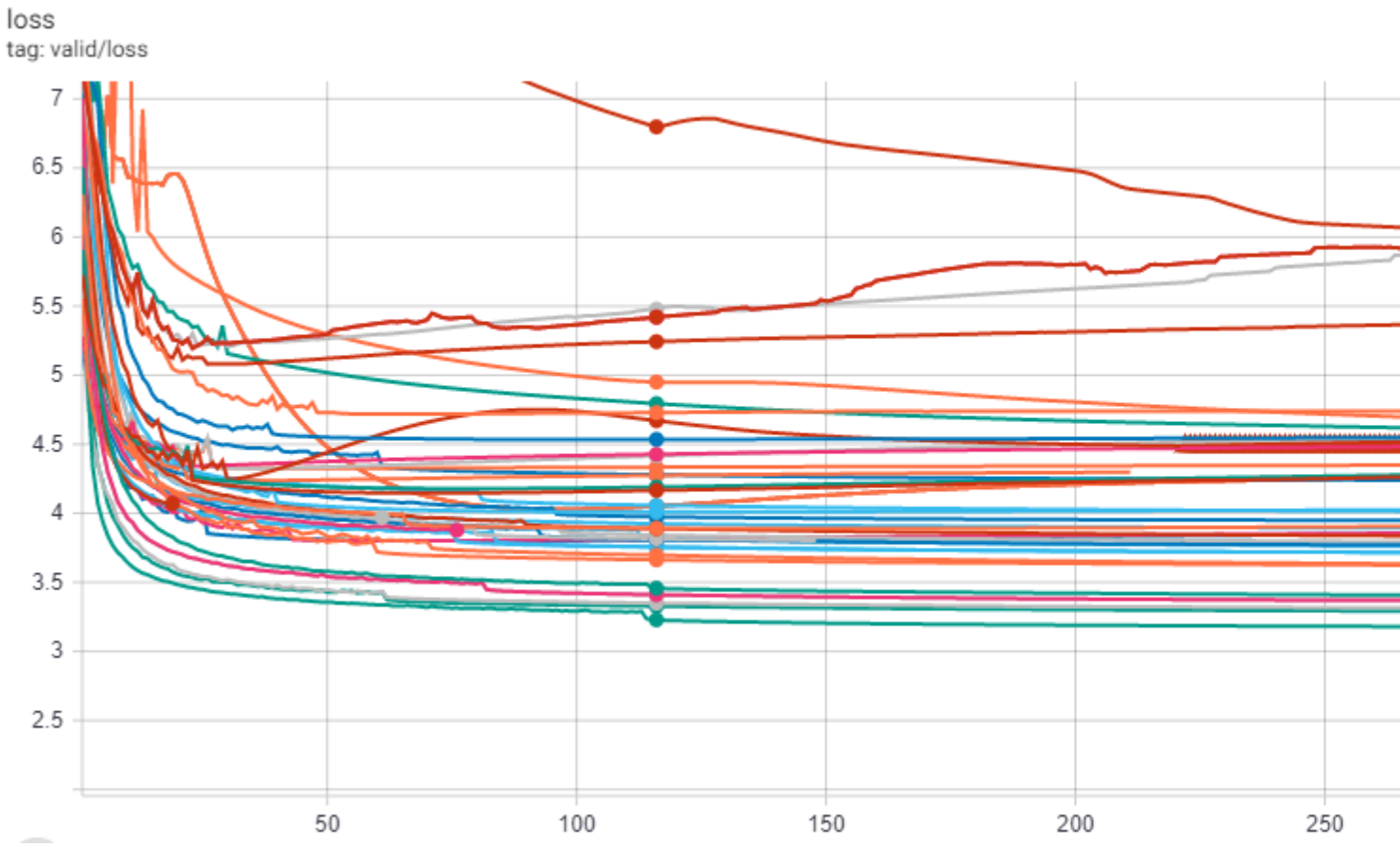
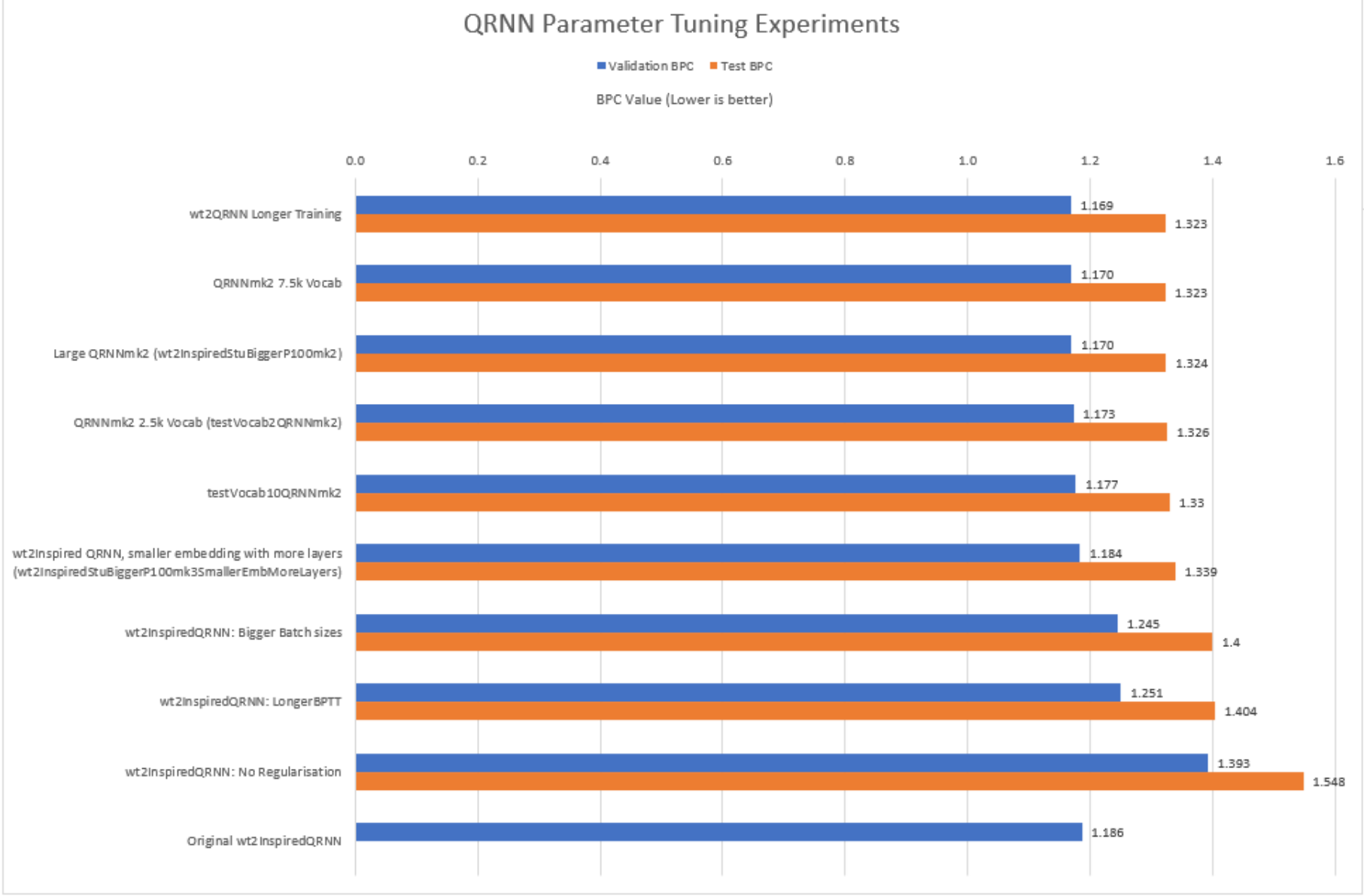


Figure 6: An illustration of the overall consistency of model performance on the validations set relative to one another beyond the 150th epoch or roughly the first 3 hours training. The lines (each representing a models performance over time) which do no fit the trend are models which overfit the data, either by design or a lack of sufficient regularisation with respect to the basic LSTM model variants.

C OVERALL RESULTS

val ord	descriptive_name	Test_B	Difference	% better t	data	model	emsize	nhid	nlayers	lr	clip	epochs	batch_	bptt	dropou	dropou	dropou	dropou	wdrop	nonmon	alpha	beta	wdecay	vocab_size	hparam/val_loss	hparam/val_bpc	Difference over best BPC	% better than best BPC
1	wt2InspiredStuBiggerP100mk2_autshumato_isizu	1.637	0	0	data/autshumato/isizulu/	QRNN	800	1550	4	30	0.25	500	40	70	0.4	0.2	0.4	0.1	0.1	8	2	1	0.0000012	5000	4.533271313	1.514269114	0	0
2	wt2InspiredStuBiggerLSTM_autshumato_isizulu	1.671	0.034	2.0769701	data/autshumato/isizulu/	LSTM	800	1150	3	30	0.25	750	80	70	0.4	0.2	0.65	0.1	0.5	8	2	1	0.0000012	5000	4.717209816	1.575710893	0.061441779	4.0575204452
3	basicDropoutOnlyVariational_autshumato_isizulu	1.798	0.161	9.8350641	data/autshumato/isizulu/	LSTM	400	1550	1	5	0	500	40	70	0.4	0	0	0	0	5	0	0	0	5000	5.079578876	1.696754575	0.182485461	12.05105678
6	basicInputDrop_autshumato_isizuluX2	1.825	0.188	11.484423	data/autshumato/isizulu/	LSTM	400	1550	1	10	0	500	40	70	0	0	0.25	0	0	5	0	0	0	5000	5.194280624	1.735068917	0.220793803	14.58127891
4	basicInputDrop_autshumato_isizulu	1.862	0.225	13.744655	data/autshumato/isizulu/	LSTM	400	1550	1	10	0	500	40	70	0	0	0.25	0	0	5	0	0	0	5000	5.194280624	1.735068917	0.220793803	14.58127891
5	basicInputDrop_autshumato_isizulu	1.862	0.225	13.744655	data/autshumato/isizulu/	LSTM	400	1550	1	10	0	500	40	70	0	0	0.5	0	0	5	0	0	0	5000	5.20870924	1.739888668	0.225619554	14.89356785
7	basicInputDrop_autshumato_isizuluD	15.22	13.583	829.74954	data/autshumato/isizulu/	LSTM	400	1550	3	5	0	500	40	70	0	0	0.5	0	0	5	0	0	0	5000	36.79358673	12.29032707	10.77605796	711.63427
1	wt2InspiredStuBiggerLSTM_autshumato_sepedi	1.577	0	0	data/autshumato/sepedi/	LSTM	800	1150	3	30	0.25	750	80	70	0.4	0.2	0.65	0.1	0.5	8	2	1	0.0000012	5000	3.601182699	1.278228045	0	0
2	wt2InspiredStuBiggerP100mk2_autshumato_sep	1.585	0.008	0.5072923	data/autshumato/sepedi/	QRNN	800	1550	4	30	0.25	500	40	70	0.4	0.2	0.4	0.1	0.1	8	2	1	0.0000012	5000	3.61878109	1.284474492	0.006246447	0.488680171
3	basicInputDrop_autshumato_sepedi	1.728	0.151	9.5751427	data/autshumato/sepedi/	LSTM	400	1550	1	10	0	500	40	70	0	0	0.25	0	0	5	0	0	0	5000	4.145668507	1.471491456	0.193263411	15.1196347
4	basicInputDrop_autshumato_sepediX2	1.733	0.156	9.8922004	data/autshumato/sepedi/	LSTM	400	1550	1	5	0	500	40	70	0	0	0.5	0	0	5	0	0	0	5000	4.178360939	1.483095527	0.204867482	16.02745948
5	basicDropoutOnlyVariational_autshumato_sepe	1.839	0.262	16.613824	data/autshumato/sepedi/	LSTM	400	1550	1	10	0	500	40	70	0.4	0	0	0	0	5	0	0	0	5000	4.541532393	1.612002373	0.333774328	26.11226763
6	basicInputDrop_autshumato_sepediD	34.56	32.983	2091.5029	data/autshumato/sepedi/	LSTM	400	1550	3	5	0	500	40	70	0	0	0.5	0	0	5	0	0	0	5000	87.89849091	31.19928169	29.92105365	2340.82281
1	wt2InspiredStuBiggerP100mk2_isolezwe_zulu	1.264	0	0	data/isolezwe/isizulu/	QRNN	800	1550	4	30	0.25	500	40	70	0.4	0.2	0.4	0.1	0.1	8	2	1	0.0000012	5000	3.362620592	1.151979804	0	0
2	wt2InspiredStuBiggerP100mk2_isolezwe_isizulu	1.264	0	0	data/isolezwe/isizulu/	QRNN	800	1550	4	30	0.25	500	40	70	0.4	0.2	0.4	0.1	0.1	8	2	1	0.0000012	5000	3.362620592	1.151979804	0	0
4	basicDropoutOnlyVariational_isolezwe_isizulu	1.491	0.227	17.958861	data/isolezwe/isizulu/	LSTM	800	1150	3	30	0.25	750	80	70	0.4	0.2	0.65	0.1	0.5	8	2	1	0.0000012	5000	3.435	1.176775862	0.024796058	2.152473326
5	basicInputDrop_isolezwe_isizulu	1.677	0.413	32.674051	data/isolezwe/isizulu/	LSTM	400	1550	1	10	0	500	40	70	0.4	0	0	0	0	5	0	0	0	5000	4.009002686	1.37342	0.221440196	19.22257623
6	basicInputDrop_isolezwe_isizuluX2	2.145	0.881	69.693367	data/isolezwe/isizulu/	LSTM	400	1550	1	10	0	500	40	70	0	0	0.25	0	0	5	0	0	0	5000	4.534039497	1.553289175	0.401309371	34.83649363
3	wt2InspiredStuBiggerLSTM_isolezwe_isizulu	2.949	1.685	133.30696	data/isolezwe/isizulu/	LSTM	400	1550	1	5	0	500	40	70	0	0	0.5	0	0	5	0	0	0	5000	5.802146912	1.987722397	0.835742593	72.54837195
1	wt2InspiredQRNNLongTrain	1.323	0	0	data/nchlt/isizulu/	QRNN	800	1550	4	30	0.25	500	40	70	0.4	0.2	0.4	0.1	0.1	8	2	1	0.0000012	10000	3.707966328	1.169039488	0	0
2	testVocab7QRNNmk2	1.323	0	0	data/nchlt/isizulu/	QRNN	800	1550	4	30	0.25	500	40	70	0.4	0.2	0.4	0.1	0.1	8	2	1	0.0000012	7500	4.003436089	1.169542313	0.000502825	0.043011806
3	wt2InspiredStuBiggerP100mk2	1.324	0.001	0.0755858	data/nchlt/isizulu/	QRNN	800	1550	4	30	0.25	500	40	70	0.4	0.2	0.4	0.1	0.1	8	2	1	0.0000012	5000	3.710558653	1.169856787	0.000817239	0.06991201
4	testVocab2QRNNmk2	1.326	0.003	0.2267574	data/nchlt/isizulu/	QRNN	800	1550	4	30	0.25	500	40	70	0.4	0.2	0.4	0.1	0.1	8	2	1	0.0000012	2500	3.175323248	1.172549844	0.003510356	0.30027694
5	testVocab10QRNNmk2	1.33	0.007	0.5291005	data/nchlt/isizulu/	QRNN	800	1550	4	30	0.25	500	40	70	0.4	0.2	0.4	0.1	0.1	8	2	1	0.0000012	10000	4.23104908	1.177128553	0.008083065	0.691941126
6	wt2InspiredStuBiggerP100mk3SmallerEmbMoreL	1.339	0.016	1.2093726	data/nchlt/isizulu/	QRNN	400	1550	6	30	0.25	500	40	70	0.4	0.2	0.4	0.1	0.1	8	2	1	0.0000012	5000	3.754822493	1.183812261	0.014772773	1.263667579
8	wt2InspiredQRNNBiggerBatch	1.339	0.016	1.2093726	data/nchlt/isizulu/	QRNN	400	1550	4	30	0.25	500	40	70	0.4	0.2	0.4	0.1	0.1	5	2	1	0.0000012	5000	3.763235331	1.186464667	0.017425179	1.49055521
13	wt2InspiredQRNNBiggerBatch	1.4	0.077	5.8201058	data/nchlt/isizulu/	QRNN	400	1550	4	30	0.25	500	80	70	0.4	0.2	0.4	0.1	0.1	5	2	1	0.0000012	5000	3.791683674	1.195433736	0.026394248	2.257772151
14	wt2InspiredQRNN_LongerBPTT	1.404	0.081	6.122449	data/nchlt/isizulu/	LSTM	800	1150	3	30	0.25	750	80	70	0.4	0.2	0.65	0.1	0.5	8	2	1	0.0000012	5000	3.801731348	1.198601484	0.029561996	2.528742237
15	basicDropoutOnlyVariational	1.479	0.156	11.791383	data/nchlt/isizulu/	LSTM	400	1150	3	30	0.25	750	80	70	0.4	0.2	0.65	0.1	0.5	5	2	1	0.0000012	5000	3.814568043	1.20264864	0.033609152	2.874337275
17	basicInputDrop_nchlt_isizuluX2	1.507	0.184	13.907785	data/nchlt/isizulu/	LSTM	800	1150	3	30	0.25	750	80	70	0.4	0.2	0.65	0.1	0.5	8	2	1	0.0000012	5000	3.877608299	1.222523808	0.05348432	4.575065304
16	basicEmbeddingDrop	1.51	0.187	14.134543	data/nchlt/isizulu/	LSTM	400	1150	3	30	0.25	500	20	70	0.4	0.25	0.4	0.1	0.5	5	2	1	0.0000012	5000	3.889286041	1.226205587	0.057166099	4.89000582
18	basicInputDrop	1.518	0.195	14.739229	data/nchlt/isizulu/	QRNN	400	1550	4	30	0.25	500	40	70	0.4	0.4	0.4	0.1	0.2	5	2	1	0.0000012	5000	3.948965311	1.245021105	0.075981617	6.493491059
20	basicNoDropouts	1.544	0.221	16.70446	data/nchlt/isizulu/	QRNN	400	1200	4	30	0.25	500	40	120	0.4	0.2	0.4	0.1	0.1	5	2	1	0.0000012	5000	3.96663928	1.250593305	0.081553817	6.976138774
19	wt2InspiredQRNN_NoRegularisation	1.548	0.225	17.006803	data/nchlt/isizulu/	LSTM	400	1550	1	10	0	500	40	70	0.4	0	0	0	0	5	0	0	0	5000	4.236155987	1.335566044	0.166526556	14.24473319
7	wt2InspiredQRNN		-1.323	-100	data/nchlt/isizulu/	LSTM	400	1550	1	10	0	500	40	70	0	0	0	0.25	0	0	0	0	0	5000	4.324448586	1.363402605	0.194363117	16.62588125
9	wt2InspiredStuBiggerP100		-1.323	-100	data/nchlt/isizulu/	LSTM	400	1550	1	10	0	500	40	70	0	0	0.5	0	0	5	0	0	0	5000	4.325352669	1.363687754	0.194648266	16.65027298
10	wt2Inspired		-1.323	-100	data/nchlt/isizulu/	LSTM	400	1550	1	10	0	500	40	70	0	0	0.25	0	0	5	0	0	0	5000	4.343053341	1.369268298	0.20022881	17.12763444
11	wt2InspiredStuBigger		-1.323	-100	data/nchlt/isizulu/	QRNN	400	1550	4	30	0.25	500	40	70	0	0	0	0	0	5	0	0	0.0000012	5000	4.417477131	1.392732501	0.223693013	19.13476964
12	ptbInspired		-1.323	-100	data/nchlt/isizulu/	LSTM	400	1550	1	10	0	500	40	70	0	0	0	0	0	5	0	0	0	5000	4.42091465	1.393816233	0.224776745	19.22747241
1	wt2InspiredStuBiggerP100mk2_nchlt_sepedi	1.421	0	0	data/nchlt/sepedi/	QRNN	800	1550	4	30	0.25	500	40	70	0.4	0.2	0.4	0.1	0.1	8	2	1	0.0000012	5000	3.282463749	1.159680486	0	0
3	basicDropoutOnlyVariational_nchlt_sepedi	1.62	0.199	14.004222	data/nchlt/sepedi/	LSTM	800	1150	3	30	0.25	750	80	70	0.4	0.2	0.65	0.1	0.5	8	2	1	0.0000012	5000	3.284854412	1.160522819	0.000842333	0.072634921
4	basicInputDrop_nchlt_sepedi	1.625	0.204	14.356087	data/nchlt/sepedi/	LSTM	400	1550	1	10	0	500	40	70	0.4	0	0	0	0	5	0	0	0	5000	3.800150871	1.342574596	0.18289411	15.77107766
5	basicInputDrop_nchlt_sepediX2	1.646	0.225	15.83392	data/nchlt/sepedi/	LSTM	400	1550	1	10	0	500	40	70	0	0	0.25	0	0	5	0	0	0	5000	3.800731421	1.342779756	0.18309927	15.78876874
6	basicInputDrop_nchlt_sepediD	1.715	0.294	20.689655	data/nchlt/sepedi/	LSTM	400	1550	1	10	0	500	40	70	0	0	0.5	0	0	5	0	0	0	5000	3.864165545	1.365190625	0.205510139	17.7212725
2	wt2InspiredStuBiggerLSTM_nchlt_sepedi		-1.421	-100	data/nchlt/sepedi/	LSTM	400	1550	3	5	0	500	40	70	0	0	0.5	0	0	5	0	0	0	5000	4.027290344	1.422821879	0.263141393	22.69085288
7	basicInputDrop_nchlt_sepediD		-1.421	-100	data/nchlt/sepedi/	LSTM	400	1550	3	5	0	500	40	70	0	0	0.5	0	0	5	0	0	0	5000	4.027290344	1.422821879	0.263141393	22.69085288



	wt2QRNN Longer Training	QRNNmk2 7.5k Vocab	Large QRNNmk2 (wt2InspiredStuBiggerP100mk2)	QRNNmk2 2.5k Vocab (testVocab2QRNNmk2)	testVocab10QRNNmk2	wt2Inspired QRNN, smaller embedding with more layers (wt2InspiredStuBiggerP100mk3SmallerEmbMoreLayers)	wt2InspiredQRNN: Bigger Batch sizes	wt2InspiredQRNN: LongerBPTT	wt2InspiredQRNN: No Regularisation	Original wt2InspiredQRNN
Test BPC	1.323	1.323	1.324	1.326	1.33	1.339	1.4	1.404	1.548	
Validation BPC	1.169	1.170	1.170	1.173	1.177	1.184	1.245	1.251	1.393	1.186
Difference over best Test BPC	0	0	0.001	0.003	0.007	0.016	0.077	0.081	0.225	
Percentage better than best Test BPC	0	0	0.07558579	0.22675737	0.529100529	1.209372638	5.82010582	6.12244898	17.00680272	

Figure A1: QRNN Parameter tuning experiments on the NCHLT-isiZulu dataset. Lower valuies indicate improved model performance. The original wt2inspired QRNN did not have test results recorded. For all models, test results were hidden until after all experiments had been run and were not factored into the training or parameter selection criteria.

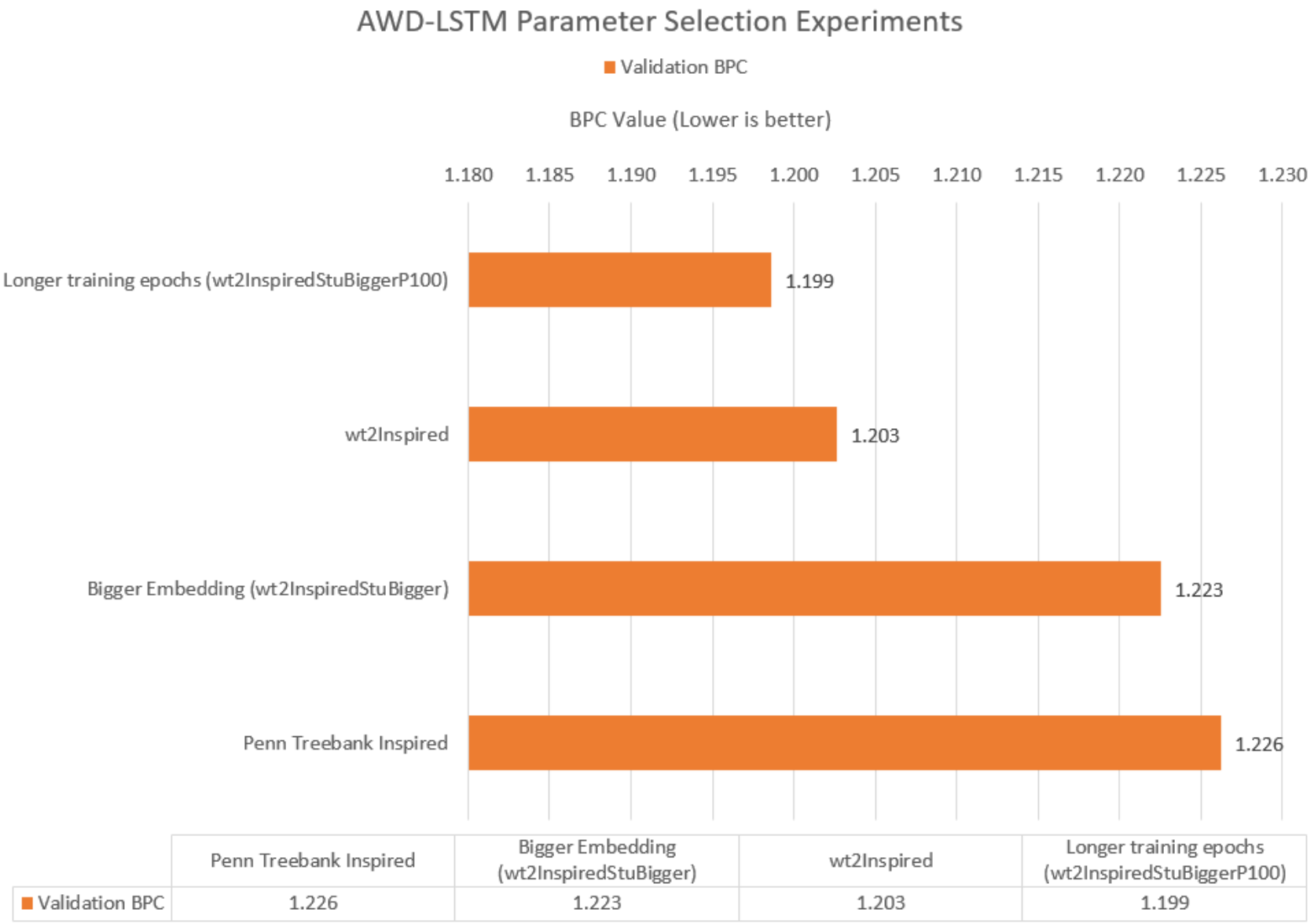
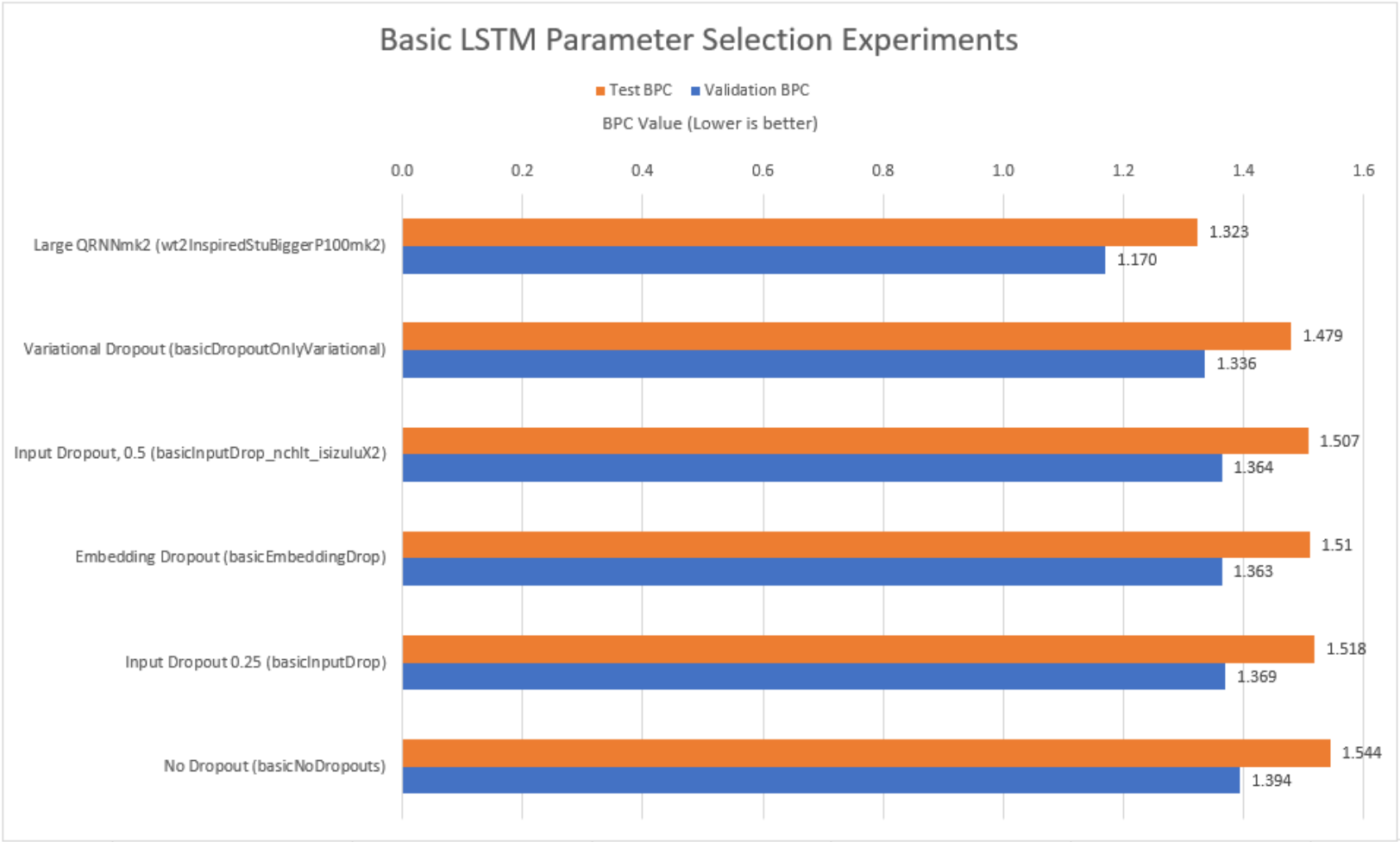
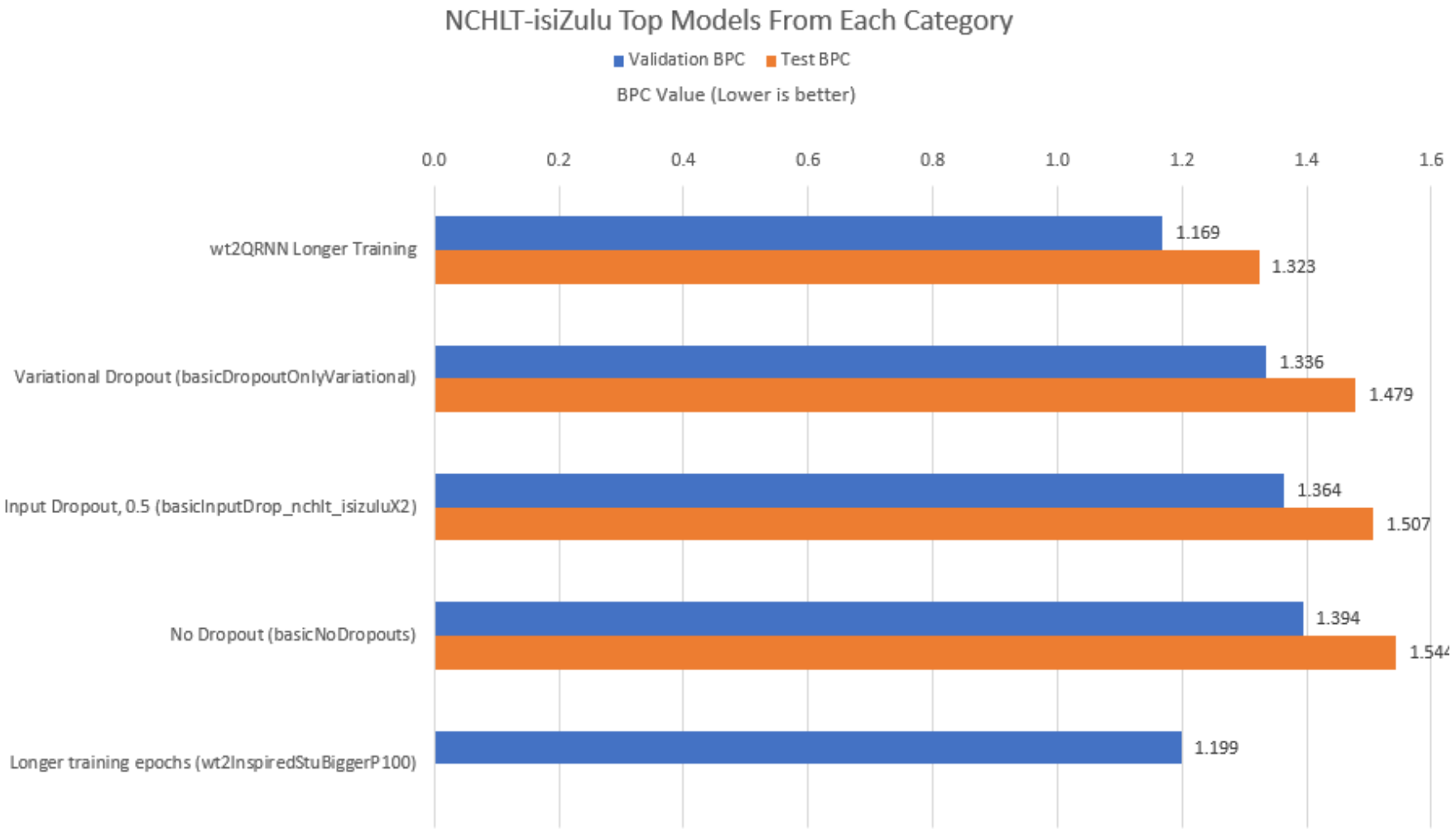


Figure A2: AWD-LSTM paramet choice on the NCHLT-isiZulu dataset. Low values indicate improved model performance. No Test set evaluation results were recorded that could be used for the reuslts tables. The larger wt2Inspire model was most effective, provided significant training time on a faster GPU was allowed for (in this



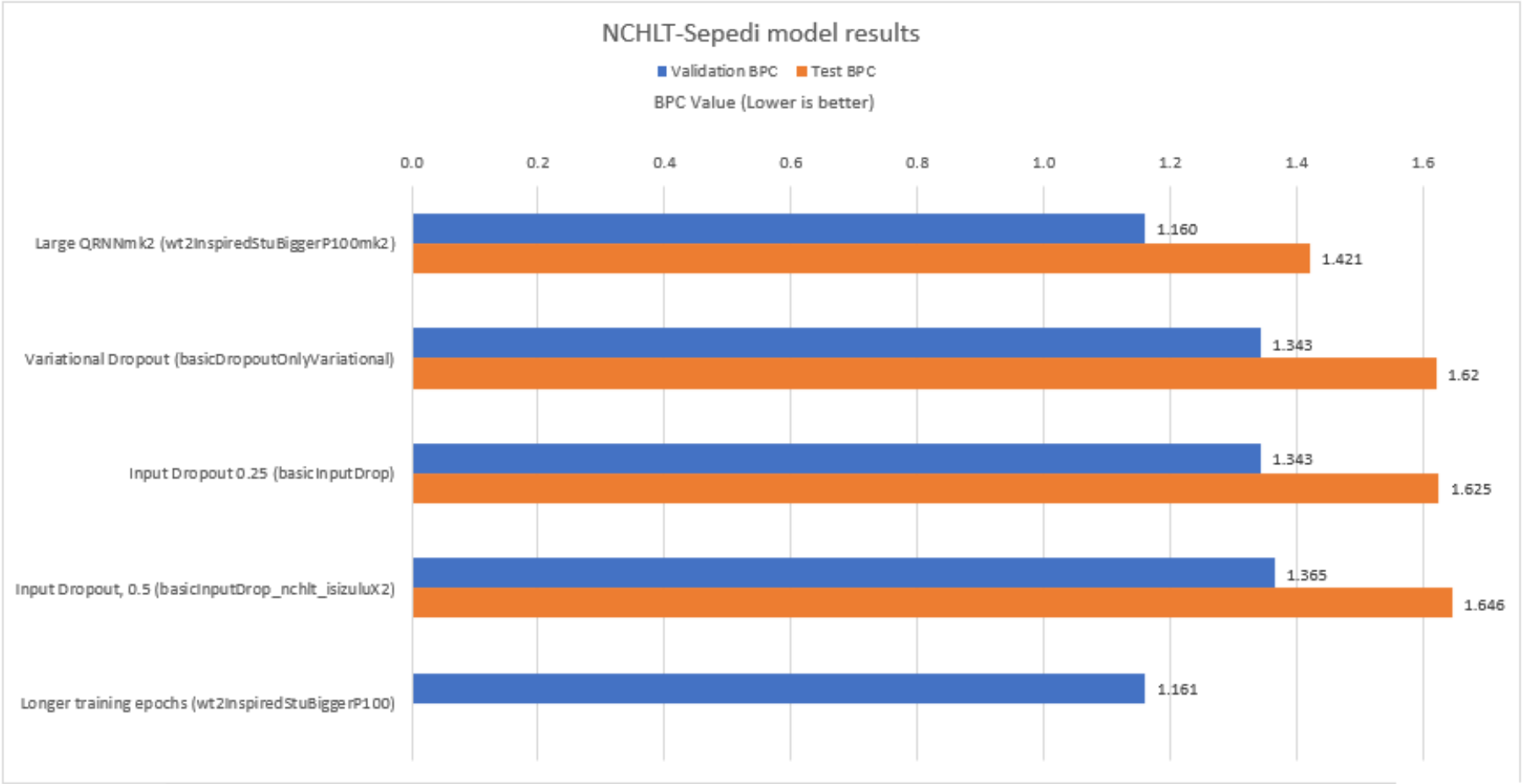
	Large QRNNmk2 (wt2InspiredStuBiggerP100mk2)	Variational Dropout (basicDropoutOnlyVariational)	Input Dropout, 0.5 (basicInputDrop_nchlt_isizuluX2)	Embedding Dropout (basicEmbeddingDrop)	Input Dropout 0.25 (basicInputDrop)	No Dropout (basicNoDropouts)
Validation BPC	1.170	1.336	1.364	1.363	1.369	1.394
Test BPC	1.323	1.479	1.507	1.51	1.518	1.544
Percentage better than best Test BPC	0	11.79138322	13.90778534	14.13454271	14.73922902	16.70445956
Difference over best Test BPC	0	0.156	0.184	0.187	0.195	0.221

Figure A3: Comparison of Basic LSTM model variants against top performing QRNN model (Top) on the NCHLT-isiZulu dataset. All basic LSTM models underperform the QRNN in both validaiton and test results.



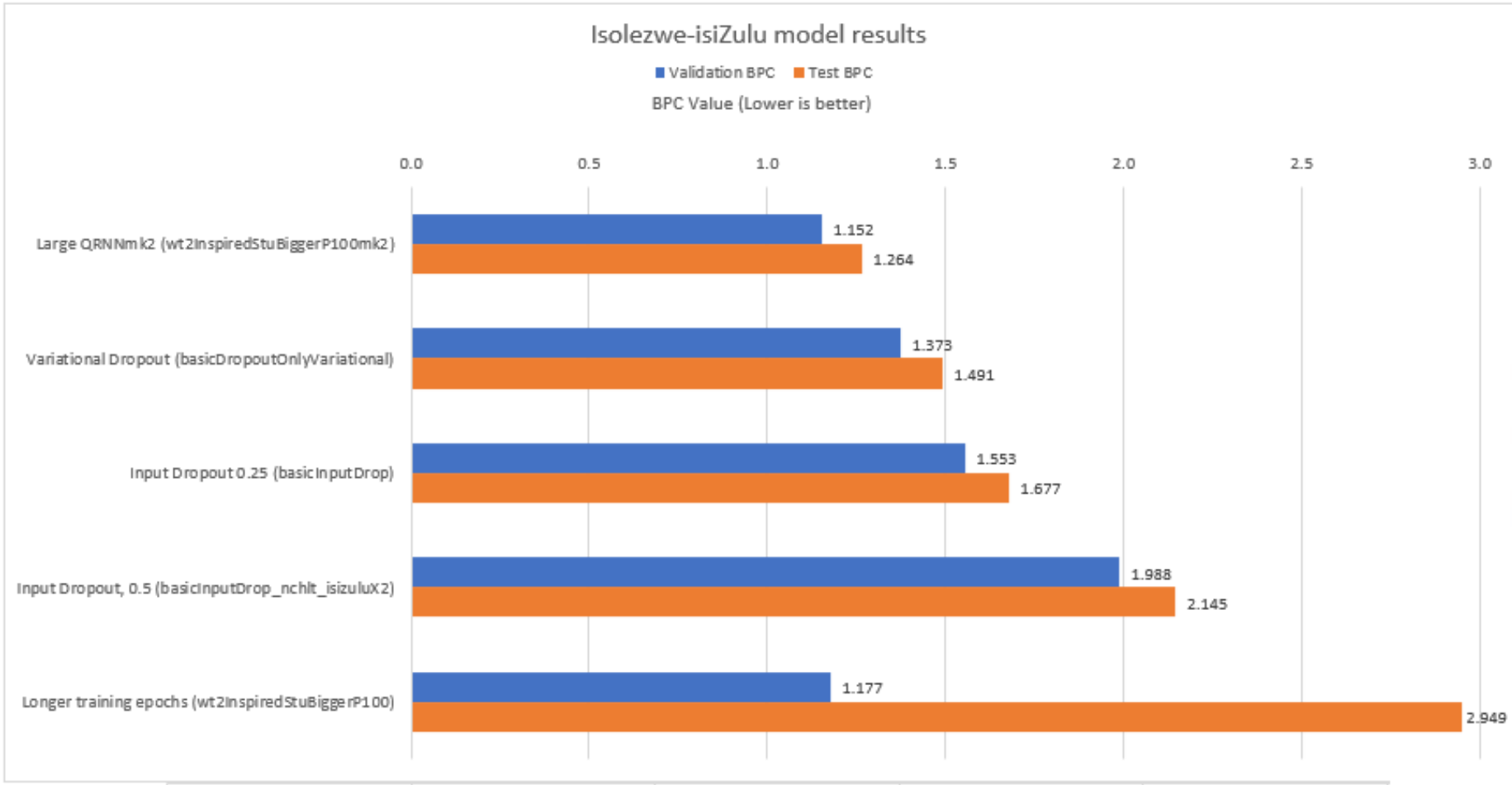
	wt2QRNN Longer Training	Variational Dropout (basicDropoutOnlyVariational)	Input Dropout, 0.5 (basicInputDrop_nchlt_isizuluX2)	No Dropout (basicNoDropouts)	Longer training epochs (wt2InspiredStuBiggerP100)
Test BPC	1.323	1.479	1.507	1.544	
Validation BPC	1.169	1.336	1.364	1.394	1.199
model	QRNN	LSTM	LSTM	LSTM	AWD-LSTM
Percentage better than best Test BPC	0	11.79138322	13.90778534	16.70445956	
Difference over best Test BPC	0	0.156	0.184	0.221	

Figure A3: Comparison of the top performing models from each category, QRNN, AWD-LSTM and basic LSTM (multiple variants).



	Large QRNNmk2 (wt2InspiredStuBiggerP100mk2)	Variational Dropout (basicDropoutOnlyVariational)	Input Dropout 0.25 (basicInputDrop)	Input Dropout, 0.5 (basicInputDrop_nchlt_isizuluX2)	Longer training epochs (wt2InspiredStuBiggerP100)
Test BPC	1.421	1.62	1.625	1.646	
Validation BPC	1.160	1.343	1.343	1.365	1.161
Percentage better than best Test BPC	0	14.00422238	14.35608726	15.83391977	
Difference over best Test BPC	0	0.199	0.204	0.225	
Model	QRNN	LSTM	LSTM	LSTM	AWD-LSTM

Figure A6: Comparison of the top performing models (chosen from the NCHLT-isiZulu results) when trained and evaluated on NCHLT-Sepedi Corpus: Large QRNN, AWD-LSTM and basic LSTM (multiple variants). The AWD-LSTM model's test set performance scores were corrupted during testing and unrecoverable for this report.
Based off the trend across other dataset experiments, whereby the relative performance across the validation sets tends to indicate a matched relative performance on the test set, it can be suggested that the AWD-LSTM model would have achieved performance between the QRNN and Variational only dropout models. However this cannot be said for certain without retesting the model.

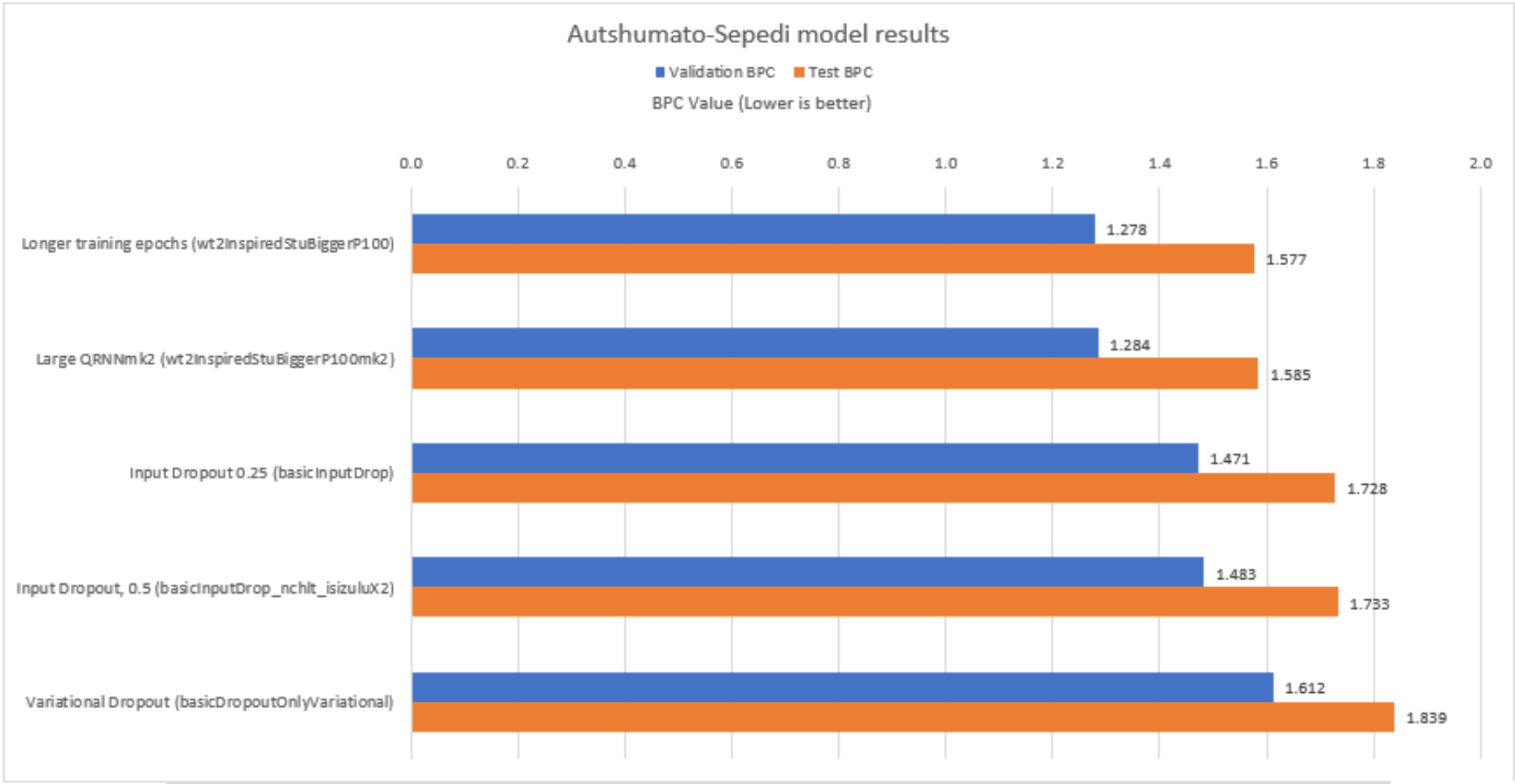


	Large QRNNmk2 (wt2InspiredStuBiggerP100mk2)	Variational Dropout (basicDropoutOnlyVariational)	Input Dropout 0.25 (basicInputDrop)	Input Dropout, 0.5 (basicInputDrop_nchlt_isizuluX2)	Longer training epochs (wt2InspiredStuBiggerP100)
Test BPC	1.264	1.491	1.677	2.145	2.949
Validation BPC	1.152	1.373	1.553	1.988	1.177
Percentage better than best Test BPC	0	17.95886076	32.67405063	69.69936709	133.306962
Difference over best Test BPC	0	0.227	0.413	0.881	1.685
Model	QRNN	LSTM	LSTM	LSTM	AWD-LSTM

Figure A5: Comparison of the top performing models (chosen from the NCHLT-isiZulu results) when trained and evaluated on isolezwe-isiZulu Corpus: Large QRNN, AWD-LSTM and basic LSTM (multiple variants).
Note the AWD-LSTM uncharacteristically performs significantly worse on the test set compared to the models validationr esults and all other models tested. It is unclear the cause of this discrepancy between the valid/test results and the subsequent departure from the trends seen in other dataset experiments.

	Large QRNNmk2 (wt2InspiredStuBiggerP100mk2)	Longer training epochs (wt2InspiredStuBiggerP100)	Variational Dropout (basicDropoutOnlyVariational)	Input Dropout, 0.5 (basicInputDrop_nchIt_isizuluX2)	Input Dropout 0.25 (basicInputDrop)
Test BPC	1.637	1.671	1.798	1.825	1.862
Validation BPC	1.514	1.576	1.697	1.740	1.735
Percentage better than best Test BPC	0	2.076970067	9.835064142	11.48442272	13.74465486
Difference over best Test BPC	0	0.034	0.161	0.188	0.225
Model	QRNN	AWD-LSTM	LSTM	LSTM	LSTM

Figure A4: Comparison of the top performing models chosen from the Autshumato-isiZulu results: Large QRNN, AWD-LSTM and basic LSTM (multiple variants).



	Longer training epochs (wt2InspiredStuBiggerP100)	Large QRNNmk2 (wt2InspiredStuBiggerP100mk2)	Input Dropout 0.25 (basicInputDrop)	Input Dropout, 0.5 (basicInputDrop_nchlt_isizuluX2)	Variational Dropout (basicDropoutOnlyVariational)
Test BPC	1.577	1.585	1.728	1.733	1.839
Validation BPC	1.278	1.284	1.471	1.483	1.612
Percentage better than best Test BPC	0	0.507292327	9.575142676	9.89220038	16.61382372
Difference over best Test BPC	0	0.008	0.151	0.156	0.262
Model	AWD-LSTM	QRNN	LSTM	LSTM	LSTM

Figure A5: Comparison of the top performing models (chosen from the NCHLT-isiZulu results) when trained and evaluated on Autshumato-Sepedi Corpus: