UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Final Paper 2020

Title:

Multilingual Transformer Models for Low-Resource South African Languages

Author: Stuart Mesham

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*	0
Total marks		**80**	**80**

Multilingual Transformer Language Models for Low-Resource South African Languages

Stuart Mesham
University of Cape Town
Cape Town, South Africa
stuart.mesham@gmail.com

ABSTRACT

Recent advances in language modelling have been achieved with large models trained on high quality datasets consisting of billions of words. However, most South African languages have limited available data. We aim to determine whether the performance of transformer language models on South African languages can be improved by utilising data from multiple related languages. We train baseline monolingual models on isiZulu and Sepedi text as well as multilingual models using data from related languages. Furthermore we evaluate the effectiveness of adding language specific weights to multilingual models. We find that multilingual models successfully improve on the performance of the monolingual models for both languages while the addition of language specific weights to multilingual models does not improve performance.

CCS CONCEPTS

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

KEYWORDS

language modelling, neural networks, low-resource languages, Transformer

1 INTRODUCTION

Language modelling has applications in many areas such as machine translation, information retrieval, voice recognition and question answering [11, 16, 26, 35, 53]. Improvements in language modelling result in improved 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 OpenAI's GPT-2 [43]. However, South African languages are low-resource and the limited availability of high-quality training data makes training large South African language models challenging.

South African languages constitute large families of closely related languages, with at least some resources available in each of the languages (see Table 2). Currently, the two families with the largest amount of quality text available are the Nguni and Sotho-Tswana families. Within the Nguni and Sotho-Tswana families, isiZulu and Sepedi are the languages with the largest amounts of text respectively. Thus, these two languages are ideal candidates for this study. To fully utilise the available data, we make use of transfer learning.

We aim to address this problem by training multilingual models on multiple related languages simultaneously. Training on multiple related languages increases the amount of data available to the model and has a regularizing effect on training [36]. This reduces overfitting, making it possible to train larger models for a given target language.

Since the languages are related but not identical, we hypothesise that it will be advantageous to use some language-specific weights in combination with weights shared across languages. This enables the model to learn both language-specific and generally applicable representations [12]. We evaluate three strategies to jointly learn language-specific and shared weights.

1.1 Contributions

Broadly, we evaluate three main strategies for modelling South African languages with transformer models:

(1) We train optimised and regularised isiZulu and Sepedi **monolingual** language models.
(2) We train improved isiZulu and Sepedi models by simply **concatenating training data** from related languages and using a multilingual learning objective.
(3) We train isiZulu models on multilingual data with **combinations of language-specific and shared weights**. Specifically, we evaluate using a modification of Soft-Decoupled Encodings [52] (SDEs) for multilingual input representation (Section 2.11) and using language-specific attention layers while sharing all other weights across languages.

We demonstrate that multilingual modelling techniques successfully improve performance over monolingual models and that the relatedness of languages used in training affects the performance. Furthermore, we investigate the effects of each of the individual components of our modified SDEs.

1.2 Overview

We start with Section 2 where we describe fundamental concepts in language modelling and neural language models relevant to our study and summarise existing work on South African language modelling in section 3. Section 4 details the procedures carried out. We report the results in Section 5 and discuss their implications and possible future work in section 6, before concluding in section 7.

2 LITERATURE REVIEW

A language model assigns a probability to a sequence of words. Given the start of a sentence, the model tries to predict the next word, assigning a probability to each of the words that could potentially follow [13].

2.1 Language Model Definition

Formally, given a sequence of n words, $W_1^n = w_1, ..., w_n$ a language model m assigns a probability $P(W_1^n)$ to the sequence. Typically,

this is achieved using the chain rule of probability as follows:

$$P(W_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)...P(w_n|w_1, w_2, ..., w_{n-1}) = \prod_{k=1}^{n} P(w_k|W_1^{k-1}). \quad (1)$$

Here the model is assigning a probability to each word given the previous words in the sentence. The probability of the sequence is given by the product of the probabilities of each word.

2.2 Model 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. For the purposes of our research, we will assess language models in isolation. Thus only intrinsic evaluation measures will be considered. The intrinsic measure most commonly used to evaluate language models is perplexity. Below, we give brief explanations of the concepts of entropy, cross-entropy and their relation to perplexity.

2.2.1 Entropy. In the field of information theory, entropy is a basic measure of information. The entropy, H, of a random variable X with sample space S and a probability distribution function p is given by the equation:

$$H(X) = -\sum_{x \in S} p(x) \log_2 p(x). \quad (2)$$

The entropy describes the average amount of information produced per event observed. Since the log in the above equation is in base 2, the amount of information is measured in bits. The equation therefore describes the average number of bits of information produced per observation of X [13, 46].

2.2.2 Cross-Entropy. To determine the entropy of a language, we would need to know its true probability distribution which in practice is not known. We only have a language model which approximates the distribution. In this scenario, the cross-entropy of the model is used. The cross-entropy of one probability distribution on another is a measure of the level of divergence of the two distributions [18, 25]. Given a sample of text, the cross-entropy of a model is estimated as

$$H(W_1^n) = -\frac{1}{n} \log_2 P(W_1^n) \quad (3)$$

with the units of information again being bits due to the log being base 2 [13]. The more accurately the model approximates the true distribution of the language, the lower the cross-entropy.

2.2.3 Bits Per Character. Since we are studying open-vocabulary models, we want the choice of tokenization to be a modelling choice. This necessitates an evaluation metric which is independent of the tokenization. To illustrate the need for such a metric, consider the effects of manipulating the byte-pair encoding (Section 2.3) vocabulary size used by a model. One of the effects of increasing this vocabulary size is that the average number of characters per token is increased due to more token join operations being carried out. This increases the amount of information produced by each token. Cross-entropy measured in bits per token is therefore not a fair metric by which to compare models using different tokenizations since some may have larger tokens than others. Instead, the metric we will be using to evaluate and compare models in this study is bits per character (BPC). This is a measure of cross-entropy which is normalised for the length of the tokens and is therefore independent of the tokenization of the text. To calculate BPC of a model on a test set W_1^n we use

$$\text{BPC} = \frac{n}{c} H(W_1^n) \quad (4)$$

where $H(W_1^n)$ is the cross-entropy as defined in equation 3, W_1^n is sequence of n tokens and c is the number of characters in W_1^n.

2.3 Sub-word Tokenization

South African languages are highly agglutinative, making whole-word tokenization sub-optimal for language modelling due to potentially large vocabulary sizes and subsequent data sparsity. By contrast, character-level tokenization requires the model to learn long sequences of tokens. To better represent the structure of the languages, we use byte-pair encoding to break words into common sub-word units.

Byte-pair encoding is a compression algorithm [17] 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 [45].

2.4 Neural Language Models

In recent years, neural language models have achieved state-of-the-art results on many language modelling benchmarks. In this section, we describe the key elements of these models.

2.4.1 One-Hot Encoding. To input tokens into a neural network, they must first be converted to vector representations. The most straightforward approach to this is one-hot encoding. All tokens in the vocabulary V are given an index from 0 to $|V| - 1$. To encode a token with index i, a vector x of length $|V|$ is constructed containing all zeros except x_i, which has the value 1.

2.4.2 Embeddings. When making a next-word prediction, the context is provided as input to the neural network in the form of continuous vector embeddings of the preceding words. More formally, an embedding matrix is applied to transform each one-hot-encoded word to a continuous vector representation. Due to the construction of the one-hot encoding, this is equivalent to a lookup of the column of the embedding matrix with the index corresponding to the word being embedded.

One of the key advantages of neural language models over n-grams is that modelling on embeddings instead of exact words allows neural models to generalise better [8]. Word-vector embeddings are created such that words with similar meanings have similar embeddings [32]. Sentences which have never been seen before can be assigned a high probability by the model if they are constructed from words with similar embeddings to those of words in a sentence that existed in the training corpus. We have discussed continuous vector embeddings for word-level models to make their

advantages more intuitively understood. Vector embeddings of sub-word tokens yield the same advantage; sub-word units may have semantic relationships which can be captured by neural models.

The architecture used by transformer language models does not intrinsically represent positional information about the tokens. Therefore, to enable the model to make use of the ordering of the input tokens we must explicitly encode the positions of the tokens as part of the input. The GPT-2 architecture we use in this study uses learned positional embeddings. Similarly to the token embeddings, a one-hot encoding of the position id of a token (typically its index in the input sequence) is transformed using a learned embedding matrix. This is shown explicitly in equation 9.

2.5 Feed-Forward Neural Networks

One of the simplest architectures for neural language models is the feed-forward neural network [8]. This architecture makes the same Markov assumption as the n-gram model in that only a limited window of the previous n words are used by the model. One of the original neural language models proposed by Bengio et al. predicts $P(w_t|w_{t-1}, ..., w_{t-n+1})$ using

$$x = (W_e u_{t-1}, W_e u_{t-2}, ..., W_e u_{t-n+1}) \tag{5}$$

$$\hat{y} = \text{softmax}(b + W_o^{(1)} x + W_o^{(2)} \tanh(d + W_h x)) \tag{6}$$

where $u_i \in \mathbb{R}^{|V|}$ is the one-hot encoding of word w_i, V is the vocabulary, $W_e \in \mathbb{R}^{m \times |V|}$, $W_h \in \mathbb{R}^{h \times mn}$, $W_o^{(1)} \in \mathbb{R}^{|V| \times mn}$, $W_o^{(2)} \in \mathbb{R}^{|V| \times h}$, $b \in \mathbb{R}^{|V|}$ and $d \in \mathbb{R}^{|V|}$ are trainable parameters and $\hat{y} \in \mathbb{R}^{|V|}$ is the predicted next word probability distribution over the vocabulary. The hidden layer size h is an adjustable hyper-parameter. Of note is how W_e is re-used to create an m dimensional vector representation of each word. W_e is a word embedding matrix where column j is a vector representation of the word with index j in the vocabulary. Note that the skip connection using $W_o^{(1)}$ may be omitted in more recent neural feed-forward models.

2.6 Recurrent Neural Language Models

Recurrent Neural Networks (RNNs) are neural network models designed to process variable length input sequences $x_1, ..., x_T$ [15]. The sequence is processed iteratively; the hidden state h_t and output $\hat{y}_t$ at timestep t are calculated using

$$h_t = f(U x_t + W h_{t-1} + b) \tag{7}$$

$$\hat{y}_t = g(V h_t + c) \tag{8}$$

where function f is a sigmoid activation funcation and g is a softmax activation function [19]. Weight matrices U, V and W as well as bias vectors b and c are trainable parameters. In the context of language models, $\hat{y}$ is typically a predicted next word probability distribution over the vocabulary [31].

2.6.1 Vanishing and Exploding Gradients. During training, the error signal must be backpropagated across the hidden states at each timestep. This requires repeated multiplication of the error signal by the same weight values which are shared across timesteps. This causes an exponential growth or decrease of the error signal resulting in highly unstable error signals at the earlier timesteps. This is known as the exploding or vanishing gradient problem and poses a challenge when attempting to train RNNs on longer sequences [9, 39]. The error signal instability at earlier timesteps reduces an RNN's ability to learn long-term dependencies.

One method of dealing with this problem is gradient clipping, where gradients with norms exceeding a given threshold are re-scaled down to that threshold [39]. Another approach, often used in conjunction with gradient clipping is using model architectures designed to reduce the problem such as gated RNNs.

2.6.2 LSTMs. Gated RNN architectures are designed to mitigate the vanishing gradient problem and improve the ability of RNNs to learn long-term dependencies by including various "gates" within the model. These gates break the chain of multiplication that otherwise occurs over successive timesteps through recurrent connections, by having an additive rather than a multiplicative relationship between the contexts of successive timesteps. The successive contexts are related by an addition operator instead of the multiplicative recurrent connection weights of the original RNN architecture. This allows the gradient to propagate across timesteps without exponential decay, thereby reducing the vanishing or exploding of the gradient [39].

Long Short-Term Memory (LSTM) models are widely used gated RNNs [20]. LSTMs use a system of memory cells and gate units to achieve the gating effect described above. In practice, LSTMs significantly improve language modelling performance over vanilla RNNs [48].

2.7 Transformer Language Models

Recurrent neural networks require the hidden states for each timestep to be sequentially computed. This is a performance bottleneck for modern hardware capable of highly parallel computing. A model architecture called a transformer network eliminates this bottleneck by relying entirely on attention mechanisms [6] instead of recurrent connections for propagating information across timesteps [51]. Attention mechanisms take all input embeddings for a sequence simultaneously and selectively weight certain features based on a learned function.

The original transformer model is used for translation and has an encoder-decoder structure [51]. For the task of language modelling, only the decoder section of the architecture is used [27]. In this section, we describe the decoder-only architecture used by GPT-2 [43]. The model uses learned embeddings of length d_{model} to represent each token.

The decoder-only architecture of GPT-2 consists of N stacked identical layers. Each layer both takes as input and produces as output a vector of length d_{model} for each timestep. On a high level, the GPT-2 architecture is

$$H^{(0)} = U W_e + S W_p \tag{9}$$

$$H^{(i)} = \text{transformer\_block}(H^{(i-1)}) \forall i \in [1, N] \tag{10}$$

$$P(u) = \text{softmax}(H^{(N)} W_e^T) \tag{11}$$

where d_t is the length of the current input, L is the maximum input length, $|V|$ is the vocabulary size, $U \in \mathbb{Z}^{d_t \times |V|}$ is the input text representation where row i is the one-hot encoding for the word at position i in the text, $S \in \mathbb{R}^{d_t \times L}$ is a position matrix where row i is

the one-hot encoding for the position of word i (S can be conceptualised as the identity matrix truncated to d_t rows), $W_e \in \mathbb{R}^{|V| \times d_{\text{model}}}$ is the word embedding matrix and $W_p \in \mathbb{R}^{L \times d_{\text{model}}}$ is the position embedding matrix, $H^{(i)} \in \mathbb{R}^{d_t \times d_{\text{model}}}$ is the output of hidden layer i and $P(u) \in \mathbb{R}^{d_t \times |V|}$ is the predicted next word probability distribution over the vocabulary. W_e and W_p are trainable weight matrices. Each row t of $H^{(i)}$ contains the output of hidden layer i at timestep t.

The structure within transformer_block [10] is

$$A^{(0)} = \text{per\_timestep\_layer\_norm}(H) \tag{12}$$

$$A^{(1)} = \text{MultiHead}(A^{(0)}, A^{(0)}, A^{(0)}) + H \tag{13}$$

$$A^{(2)} = \text{per\_timestep\_layer\_norm}(A^{(1)}) \tag{14}$$

$$\text{transformer\_block}(H) = \text{feed\_forward}(A^{(2)}) + A^{(2)} \tag{15}$$

where the per_timestep_layer_norm function applies LayerNorm [5] independently to each row (corresponding to each timestep) of its input matrix using

$$\begin{gathered}\text{per\_timestep\_layer\_norm}(X) = \text{Concat}(\text{norm}_1, ..., \text{norm}_{d_t}) \\ \text{where norm}_i = \text{layer\_norm}(X_{i*}).\end{gathered} \tag{16}$$

LayerNorm is applied to a row using

$$\text{layer\_norm}(x) = a \odot \left(\frac{x - \text{mean}(x)}{\sqrt{\text{var}(x) + \epsilon}} \right) + b \tag{17}$$

where $a, b \in \mathbb{R}^{d_m}$ are trainable weights which are not shared with subsequent applications of layer_norm [10]. Finally, the feed_forward network is applied using

$$\text{feed\_forward}(X) = \text{GELU}(XW_{\text{ff}}^{(1)} + b^{(1)})W_{\text{ff}}^{(2)} + b^{(2)} \tag{18}$$

where $W_{\text{ff}}^{(1)} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}$, $W_{\text{ff}}^{(2)} \in \mathbb{R}^{d_{\text{ff}} \times d_{\text{model}}}$, $b^{(1)} \in \mathbb{R}^{d_{\text{ff}}}$ and $b^{(2)} \in \mathbb{R}^{d_{\text{model}}}$ are trainable weights and $X \in \mathbb{R}^{d_t \times d_{\text{model}}}$. These are not shared with subsequent applications of feed_forward.

Transformer models use a multi-head attention mechanism, in which h independent attention heads learn a scaled dot-product attention function

$$\begin{gathered}\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \\ \text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)\end{gathered} \tag{19}$$

where $Q, K, V \in \mathbb{R}^{d_t \times d_{\text{model}}}$ and Q is the query matrix, K is the key matrix and V is the value matrix. The $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$ and $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ matrices are trainable parameters specific to head i. These matrices are applied to input keys, queries and values to produce outputs of dimensions W_i^Q, W_i^K and W_i^V respectively. The results of these transformations are supplied as input to the attention function (equation 20) . The learned weight matrix $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$ is applied to the concatenation of the output of each head to obtain the output of the multi-head attention function.

where the input keys, queries and values are all of dimension d_{model} ($Q, K, V \in \mathbb{R}^{d_t \times d_{\text{model}}}$). Each head h_i has its own set of trained weight matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$ and $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ which transform the keys queries and values to dimensions d_k, d_k and d_v respectively before passing them to the attention function in equation 20. Finally, the outputs of the heads are concatenated and transformed with $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$ to yield an output of dimension d_{model}, the same dimension as the input to the multi-head attention layer.

Scaled dot product attention is calculated using

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{20}$$

where, Q, K and V are matrices containing queries, keys and values for each timestep, but unlike equation 19, $Q, K \in \mathbb{R}^{d_t \times d_k}$ and $V \in \mathbb{R}^{d_t \times d_v}$. The function computes a weighted sum of V over each timestep (the rows of V). The weight given to each timestep is determined by the dot product similarity of the corresponding key and query vectors, scaled by $\frac{1}{\sqrt{d_k}}$. The transformation QK^T simultaneously computes the dot products of the key and query vectors for each timestep.

During training, each self-attention head of the decoder has access to the entirety of the target sentence, including the current word being predicted and future words. This information leak enables the model able to "cheat" and simply learn to base its prediction of the current word on the current word appearing in the training set, instead of learning to make predictions from the context words. To prevent this, any of the values of the input to the attention softmax which allow the model to "cheat" are set to $-\infty$ during training, thereby preserving the auto-regressive property of the model. Specifically, if we let $T = \frac{QK^T}{\sqrt{d_k}}$, each element $T_{ij} = \frac{Q_{i*} \cdot K_{j*}}{\sqrt{d_k}}$. Since each row Q_{r*} and K_{r*} contains information from timestep r, T_{ij} contains information from timesteps i and j. When calculating the attention at timestep t, the values T_{ij} contain illegal information where $i > t$ or $j > t$ and are therefore masked with a value of $-\infty$ resulting in a value of 0 at the illegal positions once the softmax function has been applied.

2.8 Model Optimisation and Regularisation

In a low-resource setting, a model can fit an overly complex function to the available training data, attempting to accurately describe every data point. This means the model may fit the statistical noise in the data instead of more meaningful patterns and fail to generalise to test data [49]. To prevent this, we attempt to manage the complexity of the function that the model learns.

A model sensitive to fluctuations in the data is said to have high variance. On the other hand, a model that is unable to learn sufficiently from the patterns in the data is said to have high bias [24]. Decreasing variance typically increases bias; thus a careful balance between the bias and variance of the model must be found: the model must have a low enough bias to adequately fit the data whilst not having a high enough variance to overfit.

2.8.1 Model Size. The more learnable parameters a model has, the greater its variance. Thus, one way to control the variance of a model is by tuning the number of weights in the model. This is typically done by changing the number and sizes of the hidden layers. A recent study compared the performance of different transformer network depths for the translation of South African languages [50].

Another describes a strategy for automatically tuning the size of transformer networks in the context of low-resource translation [34].

2.8.2 Optimisation. There are also many existing gradient-descent-based optimisation algorithms and they typically have numerous parameters that can be tuned. This can include parameters such as learning rate, momentum and stopping conditions [30]. Deliberate tuning of these parameters can improve model performance.

2.8.3 Regularisation. Deep neural language models are complex and generally have high variance making them prone to overfitting in low-resource settings. This effect can be mitigated using various regularisation strategies.

L1 and L2 regularisation add a loss function term to penalize large weight values, preventing the model from being overly reliant on any particular weight [37]. Another technique called dropout temporarily hides a random subset of neurons during each training step [47]. This adds noise and similarly prevents the model from being overly reliant on any particular neuron. These techniques are broadly applicable to neural networks and can be used in feed-forward neural networks, RNNs and transformers [8, 30, 51].

Weight tying shares weights across layers, reducing the number of parameters and hence reducing overfitting [22, 41]. A notable example of this is the sharing of the word embedding matrix at both input and output layers of GPT and GPT-2 [42, 43].

2.8.4 Normalisation. During training, the input distributions to each layer of a neural network change as weights in the previous layers are updated. This is called covariate shift. Instability results from weights of subsequent layers being trained on distributions which continuously shift during training. By normalising the values passing through internal layers of neural networks, the covariate shift is reduced [23]. In other words: the input distributions to intermediate layers are made more consistent. This increases stability during training and in practice results in faster training convergence. Additionally, modifications to the normalisation of self-attention layers in transformer models have been shown to improve model performance in a low-resource setting [38].

2.9 Transfer Learning

Deep neural networks learn to produce increasingly abstract representations of their input data in successive layers of the model [7]. Some of these representations may be useful for tasks other than that which the model was originally trained for.

On many domain-specific tasks, transfer learning has been shown to yield substantial performance improvements [43]. We utilize transfer learning such that information from related languages can be used to improve model performance on a specific target language. Specifically, we train a single language model on multiple related languages simultaneously. We hypothesise that due to similarities between South African languages, many learned features will be useful in modelling multiple languages, thus allowing them to be learned once and shared across all training languages, resulting in a more efficient use of the limited training data.

The transfer of information can be further utilized by sharing only some weights while keeping others language-specific.

2.10 Multilingual and Cross-Lingual Modelling

In some instances, information from high-resource languages can be transferred to low-resource languages. Cross-lingual word representations can be pre-trained using monolingual text from a high-resource language in combination with a lexicon mapping between the high and low-resource languages [3]. These pre-trained word representations can improve the performance of language models trained on limited monolingual text in the low-resource language. A similar approach is to train a single model on multiple monolingual corpora [12, 33, 40]. Alternatively, a single model can be trained on many language pairs with parallel corpora to obtain universal representations that can improve information transfer to low-resource languages [4, 12, 21].

2.11 Soft-Decoupled Encodings

Wang et al. describe an embedding architecture termed Soft-Decoupled Encodings (SDEs) designed for multilingual language modelling. SDEs use language-agnostic semantic embeddings representing universal semantic concepts in addition to language-specific lexical embeddings. The design aims to allow the model to take spelling similarities and differences between languages into account when learning universal semantic embeddings. Each input word w is converted into a bag-of-n-grams representation, $\text{BoN}(w)$. Specifically, words are converted into the set of all n-grams with $n = \{1...5\}$. This set is encoded to a frequency vector similarly to one-hot encoding as described in Section 2.4.1, but using a vocabulary V of n-grams rather than tokens and using $x_i = c$ for all n-grams appearing in w, where i is the index of the n-gram in V and c is the number of occurrences of the n-gram in w. Given an input sequence of words $w_0, ...w_{d_t}$ lexical embeddings of dimension d_{model} are calculated with

$$c(U) = \tanh(UW_c) \tag{21}$$

where each row i of $U \in \mathbb{Z}^{d_t \times |V|} = \text{BoN}(w_i)$ and $W_c \in \mathbb{R}^{|V| \times d_{\text{model}}}$ is a learned embedding matrix which is shared across languages. To enable to model to normalise for spelling differences between the languages, a language-specific fully connected layer termed the language-specific transformation is then applied with

$$e_{\text{lexical}}(U) = \tanh(c(U)W_{L_i}) \tag{22}$$

where $W_{L_i} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ is a trainable weight matrix specific to language i.

The semantic embeddings are intended to capture universal "concepts" expressed by the languages. Each token in a specific language expresses some combination of these semantic concepts and thus an attention mechanism is used to lookup a weighted sum of semantic embeddings for each input token

$$e_{\text{semantic}} = \text{Softmax}(e_{\text{lexical}}(U)W_s^T)W_s \tag{23}$$

where W_s is the trainable semantic embedding matrix. This computation is the equivalent to computing attention using the lexical embeddings $e_{\text{lexical}}(U)$ as the key and using the semantic embeddings W_s as the keys and values.

A residual connection is then made over the semantic embedding

$$e_{\text{SDE}}(U) = e_{\text{lexical}}(U) + e_{\text{semantic}}(U) \tag{24}$$

completing the definition of SDEs. These embeddings used as inputs to the model.

3 PRIOR WORK ON SOUTH AFRICAN LANGUAGE MODELLING

There is limited existing literature on neural language modelling of South African languages. One study compares the performance of order 5 n-grams and single-layer vanilla RNNs with a hidden size of 300 [44]. The study finds that n-grams are better performing than RNNs in all non-European South African languages. Other more recent studies compare transformer and convolutional models for machine translation of South African languages. The results suggested transformer models to be more effective and that the use of byte-pair encoding improves performance [1, 2, 29].

4 METHODS

In this section, we detail the experiments performed in this study to answer our three research questions:

(1) Does training transformer language models on multilingual data result in improved performance of models trained on monolingual data?
(2) Does the inclusion of language-specific attention layers improve the performance of transformer language models trained on multilingual data?
(3) Does the addition of Soft-Decoupled Encodings improve the performance of transformer language models trained on multilingual data?

All preprocessing, training and evaluation functions were built into a single repository[1]. A general training and testing harness was developed to enable systematic testing and record keeping.

4.1 Ethical Considerations

One of the main ethical issues related to language modelling is the potential for language models to be used for misinformation by the automated generation of fake news. However, it is unlikely that low-resource language models could be used to generate believable news articles, hence this issue does not apply to our project. We do not foresee any other ethical implications of the project.

4.2 Dataset Preprocessing

Monolingual text from the NCHLT text project was retrieved for each language from the online repository maintained by The South African Centre for Digital Language Resources (SADiLaR) [2]. The dataset is under the Creative Commons Attribution 2.5 South Africa License [3] which permits the creation of derivative works such as this project. Artefacts were removed such as paragraph numberings and HTML tags. Manual inspection of the data revealed the presence of "junk sentences" - sentences which were frequently repeated, and appeared to be technical instructions and warnings in both the dataset language and English, presumably from headers and footers of the source web pages. We computationally searched for and removed sentences with anomalously high frequencies. The maximum threshold for the number of times a sentence could be repeated before being considered junk was determined by computing the frequencies of all sentences and setting the maximum threshold to the sum of the median frequency and the standard deviation of the frequency. Manual inspection showed this to be an effective way to detect junk sentences in all language datasets. The total numbers of sentences remaining in each dataset after pre-processing are shown in Table 2.

It should be noted that the NCHLT datasets consist largely of articles scraped from government web-pages in different languages. This means that the datasets likely contain much of the same content translated into different languages. When performing multilingual language modelling, this introduces a potential information leak. Translated versions of parts or the whole of the test set in one language could potentially appear in the training sets of other languages, thus a multilingual model may have been trained on translated versions of the test set it is evaluated on. While we expect this effect to be small since translated versions of a text will likely not be identical, we must acknowledge that this affects the reliability of the results.

4.3 Experiment Design

The OpenAI GPT-2 [43] PyTorch[4] implementation provided by the open-source Huggingface library was used. We used the byte-pair encoding [45] implementation provided by the Huggingface[5] library. The model was optimized using the AdamW optimizer [28] implementation provided by the Huggingface library. The AdamW optimiser provides a weight decay parameter which is designed to have a regularising effect equivalent to L2 regularisation (Section 2.8.3). Weight decay and dropout (Section 2.8.3) applied to the input embeddings, the output of the softmax function in the attention heads (Section 2.7) and the output of the fully connected layers at the end of each transformer block are used to regularise the models.

We chose the NCHLT datasets for all languages used in this study. To maximise the possibility of performance gains from transfer learning, it is ideal to have datasets collected from the same domain. The NCHLT corpora are largely scraped from governmental websites where much of the same content has been translated to multiple languages. This makes the NCHLT datasets a compelling choice for the evaluation of transfer learning.

We trained and optimised a baseline monolingual isiZulu model. We then did the same for three multilingual models with isiZulu as the target language. One multilingual model was trained on languages within the Nguni language family based on the assumption that other Nguni languages would be most informative in learning isiZulu due to their relationship. To test this assumption, we train another multilingual model targeting isiZulu, but trained on isiZulu and Sotho-Tswana languages instead of the Nguni languages. We hypothesise that the model trained on Nguni languages (including isiZulu) will perform better due to the closer relationship between target and auxiliary languages than the model trained on isiZulu

[1] https://github.com/StuartMesham/low_resource_lm
[2] www.sadilar.org
[3] https://creativecommons.org/licenses/by/2.5/za/legalcode
[4] https://pytorch.org
[5] https://github.com/huggingface/tokenizers

and Sotho-Tswana languages. Finally, we train a multilingual model on all languages in the dataset. This is to account for the possibility that most or all of the non-European South African languages are related enough that they are informative in training a model with isiZulu as the target language.

We modify the Soft-Decoupled Encodings (SDEs) proposed for neural machine translation of low-resource languages [52] for use with causal language modelling and specifically with South African languages. The first modification is the use of byte-pair encoding (BPE) sub-word tokens instead of the bag-of-n-grams (BoN) method originally proposed. This is motivated by the agglutinative nature of South African languages. We hypothesise that BPE sub-words more explicitly model the independent morphemes constituting words in South African languages. The second modification we make is to add a second language-specific transformation in the form of a feed-forward layer before the prediction head of the model. This is to account for the fact that SDEs were originally proposed in the context of neural machine translation where an encoder-decoder architecture is used. The encoder outputs a language-agnostic embedding of the encoded text which may be decoded by a language-specific decoder. This differs from our causal language model where the model must output language-specific token probabilities. We therefore hypothesise that a language-specific transformation here enables the model to produce language-agnostic representations internally and have our final language-specific transformation account for minor differences such as spelling differences between languages. We refer to this in Section 5.3 as the **output language-specific transformation**.

4.4 Hyper-Parameter Tuning

For each of the mentioned experiments, hyper-parameters were tuned over many training runs. The training time varied depending on the hyperparameters used with the larger models taking up to 16 hours to train on an Nvidia V100 GPU. The training runs were limited 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. In all cases where training was stopped early, manual inspection of the training graphs of the train and validation losses showed that the model had overfit the training set.

The training data was input to the model in **blocks** of **128** consecutive tokens in **batches** of **32** input blocks. The input blocks were created using a sliding window strategy 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** over the supplied test data.

The tuning was systematically performed, starting with smaller model sizes and smaller vocabulary sizes. For each model and vocabulary size, first little regularization was applied to see whether the model had enough capacity to overfit the data. If not, the model size was increased. Increasing amounts of regularization were then applied in successive runs. As regularization increased, the performance would increase until the model no longer overfit the data. After this point increasing the regularization reduced performance as the model would then underfit.

Preliminary testing showed that the model was relatively insensitive to the number of hidden layers and the number of attention heads. All training runs thereafter used **8 hidden layers** and **8 attention heads**. The **learning rate** was also varied in preliminary tests and an initial rate of $\mathbf{1 \times 10^{-4}}$ with a **schedule** that **linearly** decreased to 0 over the course of the training was found to have satisfactory behaviour on models of widely differing sizes and was therefore used in all subsequent training runs.

For the monolingual models, only the hidden size, vocabulary size, dropout probability and weight decay parameters were fine-tuned. The optimal parameters found for the monolingual model (Table 4a) were used to inform the starting set of parameters in optimizing the multilingual models (Table 4b and Table 4c).

For multilingual models, the tokenizer training data used was treated as a modelling decision and was fine-tuned along with other hyper-parameters. Specifically, for multilingual models, we tested selecting different subsets of the training data to train the byte-pair encoding (BPE) vocabulary to optimise performance. For the models trained on data from languages within the same family, we performed preliminary testing training the tokenizer on all languages within the family and on only the target language. Similarly, for multilingual models trained on all languages, we tested training the tokenizer on all languages, only the language family and only the target language. In our preliminary tests, model performance was found to be relatively insensitive to these differences in tokenizer training data. Tokenization training sets with the best performance in these tests were used for all remaining tests.

5 RESULTS

Here we describe the results of our main experiments. Sections 5.1-5.3 address research questions 1-3 respectively. Starting in Section 5.1 we compare the multilingual models with varying auxiliary datasets against a monolingual baseline. Following this, in Section 5.2 we compare models with language family-specific attention layers against a baseline with no family-specific attention layers. Finally, in Section 5.3 we compare models using variants of Soft-Decoupled Encodings against a baseline with no family-specific weights.

5.1 Shared Parameter Models

A total of 8 models were trained. For each of isiZulu and Sepedi we trained four models: one monolingual baseline, and three multilingual with differing auxiliary data. The three multilingual models were trained with different multilingual datasets: same-family, different-family and all-languages auxiliary data respectively. The "same-family" experiments train isiZulu and Sepedi models with Nguni and Sotho-Tswana data respectively. The "different-family" experiments swap this pairing such that isiZulu is trained on Sotho-Tswana data. Finally, the "all-languages" experiment trains isiZulu and Sepedi models on all 9 languages in our collective dataset. The hyper-parameters for each of these models were tuned independently.

Table 1, Figure 1 and Figure 2 all show different attributes of the same 8 models. Each model's p_drop, weight_decay, tokenizer training set choice, vocab_size and hidden size (d_model) were optimised independently on its target language's validation set. For

Table 1: A table showing the bits-per-character (BPC) scores of isiZulu and Sepedi transformer language models on the isiZulu and Sepedi test sets respectively. We show models trained on monolingual text, text from all languages within the language family of the target language, text from languages in a different family and text from all non-European South African languages. The sentences column shows the number of sentences (in thousands) in the training data. The tokenizer training data column shows the dataset used to train the byte-pair encoding tokenizer. Note that the tokenizer training data is always a weak subset of the model training data so that no information is introduced from outside the model training data. The model hyper-parameters were optimized on validation sets for the target languages.

Training Data	Sentences	Hyper-Parameters			Target	BPC
		Tokenizer Training Data	vocab_size	d_model		
isiZulu	96k	isiZulu	8000	256	isiZulu	1.391
Nguni Languages	291k	Nguni Languages	8000	512		1.334
isiZulu & Sotho-Tswana Languages	282k	isiZulu & Sotho-Tswana Languages	8000	512		1.331
All Languages	580k	Nguni Languages	8000	968		1.298
Sepedi	79k	Sepedi	2000	256	Sepedi	1.495
Sotho-Tswana Languages	186k	Sotho-Tswana Languages	2000	360		1.447
Sepedi & Nguni Languages	370k	Sepedi & Nguni Languages	8000	512		1.477
All Languages	580k	Sotho-Tswana Languages	8000	800		1.416

all models, p_drop = 0.3 and weight_decay = 0.2 were found to be best performing. For the remaining hyper-parameters, the optimal values for each model are shown in Table 1. For each model, the BPC score obtained on the target language's test set are also shown in Table 1.

The isiZulu monolingual model achieved a BPC of 1.391. The multilingual isiZulu models trained on the Nguni and combined isiZulu and Sotho-Tswana datasets achieved BPCs of 1.334 and 1.331 respectively. The multilingual isiZulu model trained on all languages achieved a BPC of 1.298.

The isiZulu and Sepedi All-Language models achieved BPCs of 1.298 and 1.416 respectively, outperforming the monolingual models which achieved BPCs of 1.391 and 1.495 respectively. As more training data is introduced, the optimal model hidden sizes increase and the performance increases. For example, the isiZulu optimal monolingual model's hidden size is 256, whereas the All-Language model's hidden size is 968. The increase in optimal model capacity (dependent on hidden size) reflects the increased information provided by the auxiliary data. The corresponding improvement in test BPC shows that the additional information is informative for modelling the target language.

Figure 1 compares the effects of different auxiliary datasets on the performance of isiZulu and Sepedi language models. The isiZulu models perform similarly when trained with auxiliary data from the same family to when trained with data from a different family. There is no performance improvement when using same-family auxiliary data instead of different-family auxiliary data. By contrast, the Sepedi models trained with same-family data did perform better than those trained with different-family data. Additionally, this performance improvement is despite the fact that the same-family auxiliary data, containing 186k sentences, is smaller than the different-family data, containing 370k sentences. This shows that the Sepedi language model is able to exploit structural similarities with other Sotho-Tswana languages better than similarities with Nguni languages. Finally, for both target languages, the models trained with all-language auxiliary data are the best performing.

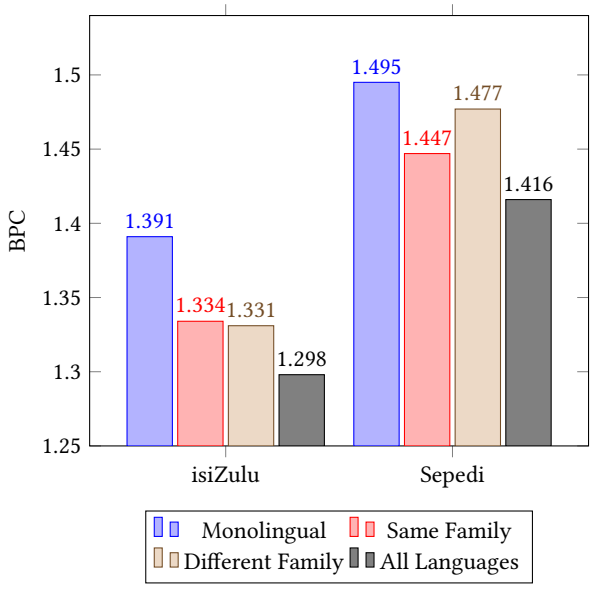

Figure 1: A bar graph showing the bits-per-character (BPC) scores obtained by 8 different language models on isiZulu and Sepedi test sets. Lower scores indicate better performance. Models were trained on monolingual text, text from languages in the same family, text from languages in different families and text from all 9 non-European South African languages.

It is clear from Figure 1 that adding more auxiliary training data improves performance. For both isiZulu and Sepedi, the models trained on all languages are best performing and the models trained on "Same Family" and "Different Family" data both outperform the monolingual models. Figure 2 shows the relationship between training sentences and test BPC for the same experiments shown in Figure 1. There is a roughly linear relationship between training sentences and test BPC despite different languages being added in

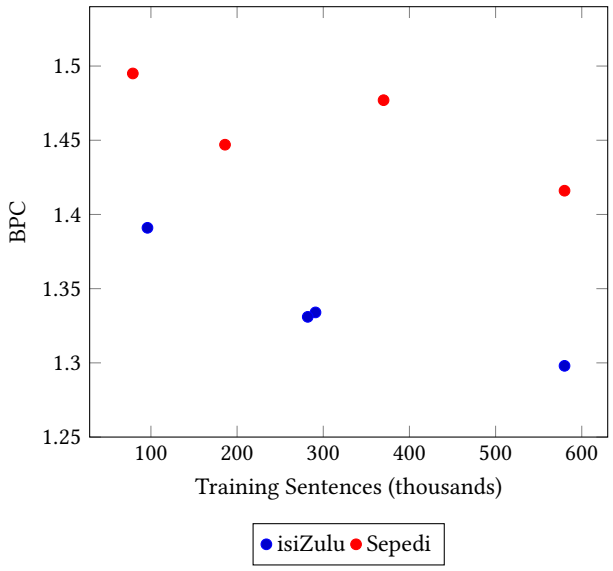

Figure 2: A scatter plot showing the test bits-per-character (BPC) scores of isiZulu and Sepedi language models trained on datasets of varying sizes. Lower scores indicate better performance. The dataset size is not the only variable being manipulated; the datasets also incorporate text from different languages as the size varies, but this is not shown. This plot serves to illustrate the strength of the effect of dataset size on performance even when it is less related languages which are being added to or removed from the corpus.

each experiment. The Sepedi experiments show greater deviation from a straight line due to the relatedness of Sepedi with other Sotho-Tswana languages as previously discussed.

5.2 Language-Specific Attention Layers

Figure 4 shows the results of training isiZulu models on multilingual data with varying numbers of language or family-specific attention layers. All models used p_drop = 0.3, weight_decay = 0.2, vocab_size = 8000 and used a tokenizer trained on all Nguni languages. The models trained with only Nguni auxiliary data used d_model = 512 whereas those trained with all language auxiliary used d_model = 800. We hypothesised that the addition of language-specific attention layers would improve performance over a multilingual baseline. However, the results show no performance improvement. Adding language-specific layers had either negligible or slightly detrimental effects on performance. One potential explanation for this is that there may be relatively similar grammar and word-orderings between the languages, making language-specific attention heads redundant and requiring each to independently learn the same function. However, our results do not contain the information required to support this hypothesis, thus more targeted experiments would be required to investigate this further.

5.3 Soft-Decoupled Encodings

In Figure 3 we compare the performance of different variants of Soft-Decoupled Encodings (SDEs) against a control model. All

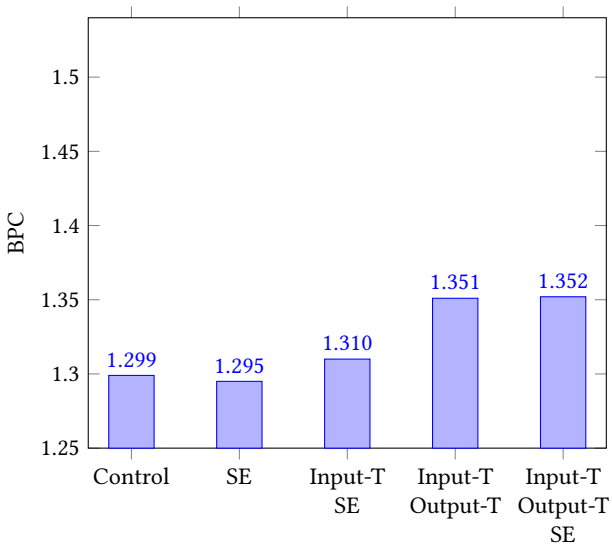

Figure 3: A bar graph showing isiZulu test bits-per-character (BPC) scores of four different multilingual model with Soft-Decoupled Encoding variants and one control model with no family specific weights. Lower scores indicate better performance. All models were trained on all non-European South African languages.

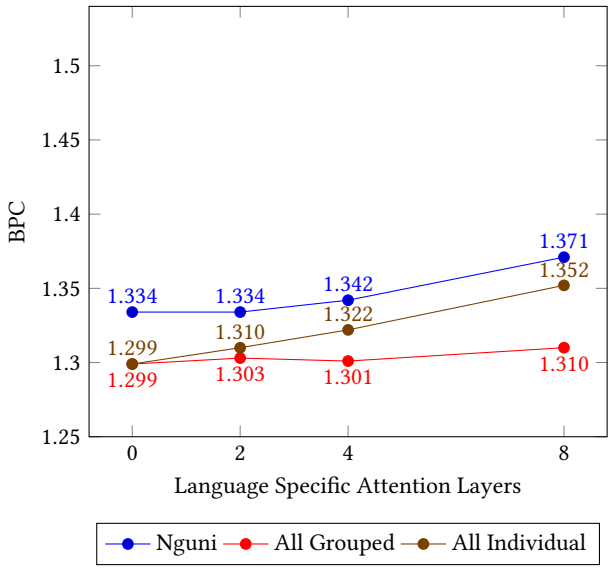

Figure 4: A line graph showing the isiZulu test bits-per-character (BPC) scores of multilingual transformer language models trained on different datasets with the bottom x attention layers being language specific. Lower scores indicate better performance. "All" denotes all non-European South African languages. Each model had 8 attention layers in total. In the "all grouped" series, languages were grouped into their families making the attention layers "family specific" rather than language specific.

models were trained on all languages and used p_drop = 0.3, weight_decay = 0.2, d_model = 800, vocab_size = 8000 and used a tokenizer trained on all Nguni languages. Our SDE variants are constructed from different combinations of:

(1) Input family-specific transformations ("Input-T", Sec. 2.11).
(2) Output family-specific transformations ("Output-T", Sec. 4.3).
(3) Shared semantic embeddings ("SE", Sec. 2.11).

We hypothesised that the SDEs would improve performance due to the input and output transformations normalising for spelling differences and the semantic embeddings learning universal semantic concepts. However, the results do not show a significant improvement over the baseline model. The "SE" model scores approximately the same as the baseline "Control" model with a 0.004 BPC difference between the two. We also compare the "Input-T Output-T SE" and "Input-T Output-T" which also shows the addition of the semantic embeddings to have had a negligible effect with a difference of 0.001 BPC between the two. By comparing the "SE", "Input-T SE" and "Input-T Output-T SE" models sequentially we see that both the addition of the input transformation and the addition of the output transformation reduce performance by 0.015 and 0.042 BPC respectively.

A possible reason for the ineffectiveness of SDEs may be that the sub-word vocabulary and spellings may be highly similar across the families, making the family-specific transformations redundant similar to the potential issue discussed in Section 5.2. Once again, a targeted experiment would have to be performed to investigate this.

6 DISCUSSION AND FUTURE WORK

Our results have shown that the South African languages are sufficiently related in such a way that language modelling performance for a given target language can be improved by including auxiliary training data from other languages. In our experiments, adding more training data always increased performance regardless of the language family of the additional data. This shows that there exist structural similarities between all of the non-European South African languages which can be effectively exploited by language models to improve performance.

The languages investigated in this study form part of the larger family of Southern Bantu languages [14]. Future work could investigate whether adding additional auxiliary data from the broader Southern Bantu family can further improve language modelling performance. By extension, future studies could attempt to determine the limit of this strategy, by continuing to expand to broader language families until there is no longer an improvement in performance.

Our results also showed that our language model could better exploit the structural and lexical similarities between Sepedi and Sotho-Tswana languages than Sepedi and Nguni languages. Future work could search for other such language combinations since this could potentially reveal factors influencing the effectiveness of auxiliary languages in improving performance.

Both of our experiments in adding language-specific weights did not yield any performance improvements. Future work could investigate the use of techniques that more explicitly model relationships across languages such as cross-lingual word embeddings [3].

6.1 Limitations

Model training is computationally intensive. This meant that large models trained in Sections 5.1, 5.2 and 5.3 could not be thoroughly tuned. It is possible that with more tuning the results may have been improved. Additionally, in some instances such as the multilingual models trained on all languages, larger models may have achieved better performance, but were not tested due to their high computational requirements. This, as well as the potential test set information leak discussed in Section 4.2 limit the reliability of the results.

7 CONCLUSIONS

In conclusion, in the context of modelling low-resource South African languages, we have answered our three research questions. Training transformer language models on multilingual data does improve performance over models trained on monolingual data. For Sepedi models, we find that using auxiliary data from Sotho-Tswana languages yields greater performance improvements than using data from Nguni languages. For isiZulu models, we find that Nguni and Sotho-Tswana auxiliary data both yield the same performance improvement. For both isiZulu and Sepedi, models trained on all 9 non-European South African languages were the best performing, with BPC scores of 1.298 and 1.416 respectively.

By contrast, neither the use of family-specific attention layers nor Soft-Decoupled Encodings improved performance over a multilingual baseline with no language or family-specific weights.

Nonetheless, this work shows that the performance of language models for low-resource South African languages can be improved by training on data from other related languages.

ACKNOWLEDGMENTS

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] Jade Abbott and Laura Martinus. 2019. Benchmarking Neural Machine Translation for Southern African Languages. In *Proceedings of the 2019 Workshop on Widening Natural Language Processing*. Association for Computational Linguistics, Florence, Italy, 98–101.
[2] Jade Z Abbott and Laura Martinus. 2018. Towards neural machine translation for African languages. *Neural Information Processing Systems 2018 Workshop on Machine Learning for the Developing World* (2018).
[3] Oliver Adams, Adam Makarucha, Graham Neubig, Steven Bird, and Trevor Cohn. 2017. Cross-Lingual Word Embeddings for Low-Resource Language Modeling. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, 937–947. https://www.aclweb.org/anthology/E17-1088
[4] Mikel Artetxe and Holger Schwenk. 2019. Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond. *Transactions of the Association for Computational Linguistics* 7, 0 (2019), 597–610. https://transacl.org/index.php/tacl/article/view/1742
[5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).
[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

[7] Yoshua Bengio. 2012. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML workshop on unsupervised and transfer learning*. 17–36.

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

[9] Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* 5, 2 (1994), 157–166.

[10] Jan Buys. 2020. The GPT-2 Transformer. (2020). work in progress.

[11] 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

[12] Alexis Conneau and Guillaume Lample. 2019. Cross-lingual Language Model Pretraining. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 7059–7069. http://papers.nips.cc/paper/8928-cross-lingual-language-model-pretraining.pdf

[13] Jurafsky Daniel and James H Martin. 2019. N-gram Language Models. In *Speech and Language Processing* (3 ed.). Chapter 3.

[14] Clement M Doke. 2017. *The Southern Bantu Languages: Handbook of African Languages*. Vol. 19. Routledge.

[15] Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science* 14, 2 (1990), 179–211.

[16] Alexander Franz and Brian Milch. 2002. Searching the Web by Voice. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 2* (Taipei, Taiwan) *(COLING '02)*. Association for Computational Linguistics, USA, 1–5. https://doi.org/10.3115/1071884.1071887

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

[18] I. J. Good. 1956. Some Terminology and Notation in Information Theory. *Proceedings of the IEE - Part C: Monographs* 103, 3 (1956), 200–204.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. Sequence Modelling: Recurrent and Recursive Nets. In *Deep Learning*. MIT Press, Chapter 10, 374. http://www.deeplearningbook.org/contents/rnn.html

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

[21] Haoyang Huang, Yaobo Liang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, and Ming Zhou. 2019. Unicoder: A Universal Language Encoder by Pre-training with Multiple Cross-lingual Tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 2485–2494. https://doi.org/10.18653/v1/D19-1252

[22] Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462* (2016).

[23] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Francis Bach and David Blei (Eds.), Vol. 37. PMLR, Lille, France, 448–456. http://proceedings.mlr.press/v37/ioffe15.html

[24] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. The Bias-Variance Trade-Off. In *An introduction to statistical learning*. Vol. 112. Springer, Chapter 2.2.2, 33–42.

[25] S. Kullback and R. A. Leibler. 1951. On Information and Sufficiency. *The Annals of Mathematical Statistics* 22, 1 (1951), 79–86.

[26] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48* (New York, NY, USA) *(ICML'16)*. JMLR.org, 1378–1387.

[27] Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating Wikipedia by Summarizing Long Sequences. In *International Conference on Learning Representations*.

[28] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

[29] Laura Martinus and Jade Z Abbott. 2019. A Focus on Neural Machine Translation for African Languages. *arXiv preprint arXiv:1906.05685* (2019).

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

[31] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*. https://www.isca-speech.org/archive/archive_papers/interspeech_2010/i10_1045.pdf

[32] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 3111–3119.

[33] Phoebe Mulcaire, Jungo Kasai, and Noah A. Smith. 2019. Polyglot Contextual Representations Improve Crosslingual Transfer. 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)*. Association for Computational Linguistics, Minneapolis, Minnesota, 3912–3918. https://doi.org/10.18653/v1/N19-1392

[34] Kenton Murray, Jeffery Kinnison, Toan Q. Nguyen, Walter Scheirer, and David Chiang. 2019. Auto-Sizing the Transformer Network: Improving Speed, Efficiency, and Performance for Low-Resource Machine Translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*. Association for Computational Linguistics, Hong Kong, 231–240. https://doi.org/10.18653/v1/D19-5625

[35] 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.

[36] Graham Neubig and Junjie Hu. 2018. Rapid Adaptation of Neural Machine Translation to New Languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 875–880. https://doi.org/10.18653/v1/D18-1103

[37] Andrew Y. Ng. 2004. Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance. In *Proceedings of the Twenty-First International Conference on Machine Learning* (Banff, Alberta, Canada) *(ICML '04)*. Association for Computing Machinery, New York, NY, USA, 78. https://doi.org/10.1145/1015330.1015435

[38] Toan Q Nguyen and Julian Salazar. 2019. Transformers without tears: Improving the normalization of self-attention. *International Workshop on Spoken Language Translation* (2019).

[39] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International conference on machine learning*. 1310–1318.

[40] Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How Multilingual is Multilingual BERT?. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 4996–5001. https://doi.org/10.18653/v1/P19-1493

[41] Ofir Press and Lior Wolf. 2017. Using the Output Embedding to Improve Language Models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. 157–163.

[42] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. (2018). https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf

[43] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019). https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

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

[45] 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.

[46] C. E. Shannon. 1948. A Mathematical Theory of Communication. *Bell System Technical Journal* 27, 3 (jul 1948), 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

[47] 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.

[48] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*.

[49] Igor V Tetko, David J Livingstone, and Alexander I Luik. 1995. Neural network studies. 1. Comparison of overfitting and overtraining. *Journal of chemical information and computer sciences* 35, 5 (1995), 826–833.

[50] Elan van Biljon, Arnu Pretorius, and Julia Kreutzer. 2020. On optimal transformer depth for low-resource language translation. In *AfricaNLP Workshop 2020*.

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *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. http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[52] Xinyi Wang, Hieu Pham, Philip Arthur, and Graham Neubig. 2019. Multilingual Neural Machine Translation With Soft Decoupled Encoding. In *International*

Conference on Learning Representations.

[53] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR* abs/1609.08144 (2016). http://arxiv.org/abs/1609.08144

A SUPPLEMENTARY INFORMATION

Table 2: A table showing the number of sentences (in thousands) in each NCHLT language dataset after pre-processing as well as the totals for each language family and the combined total for the entire dataset.

Family	Language	Sentences	Family Sentences	Total Sentences
Nguni	isiZulu	96k	291k	580k
	isiXhosa	74k		
	Siswati	61k		
	isiNdebele	59k		
Sotho-Tswana	Sepedi	79k	186k	
	Sesotho	60k		
	Setswana	47k		
Tswa-Ronga	Xitsonga	57k	57k	
Venda	Tshivenda	45k	45k	

Table 3: A table showing the validation BPC of training runs performed to tune model hyper-parameters. Models were trained for 200k training steps. The size of the hidden layer tabulated under d_model. The dropout probability applied to the whole model is tabulated under p_drop. The weight decay argument of the AdamW optimizer [28] is shown as weight_decay. During training, the model was evaluated every 5k steps. If a model failed to improve after any 4 consecutive evaluations the training was stopped. The "Stopped Early" column shows whether or not a the model's training was stopped by this mechanism. The BPC column shows the bits per character score of the model on the NCHLT Sepedi validation set. All training runs used 8 hidden layers with 8 attention heads.

(a) Hyper-parameter tuning for Sepedi model trained on Sepedi and Nguni family text.

Hyper-Parameters					Metrics	
d_model	p_drop	weight_decay	tokenizer_dataset	vocab_size	BPC	Stopped Early
512	0.3	0.2	sepedi	2000	1.233	No
			sepedi & nguni	2000	1.219	No
			sepedi & nguni	8000	1.190	No

(b) Hyper-parameter tuning for Sepedi model trained on Sotho-Tswana text.

Hyper-Parameters					Metrics	
d_model	p_drop	weight_decay	tokenizer_dataset	vocab_size	BPC	Stopped Early
360	0.3	0.2	Sotho-Tswana	2000	1.168	No
512				2000	1.156	Yes
512				8000	1.168	Yes

(c) Hyper-parameter tuning for Sepedi model trained on Sepedi text.

Hyper-Parameters					Metrics	
d_model	p_drop	weight_decay	tokenizer_dataset	vocab_size	BPC	Stopped Early
256	0.3	0.2	Sepedi	2000	1.211	No
				8000	1.222	Yes

Table 4: A table showing the validation BPC of training runs performed to tune model hyper-parameters. Models were trained for 200k training steps. The size of the hidden layer tabulated under d_model. The dropout probability applied to the whole model is tabulated under p_drop. The weight decay argument of the AdamW optimizer [28] is shown as weight_decay. During training, the model was evaluated every 5k steps. If a model failed to improve after any 4 consecutive evaluations the training was stopped. The "Stopped Early" column shows whether or not a the model's training was stopped by this mechanism. The BPC column shows the bits per character score of the model on the NCHLT isiZulu validation set. All training runs used 8 hidden layers with 8 attention heads.

(a) Hyper-parameter tuning for isiZulu model trained only on isiZulu data.

Hyper-Parameters				Metrics	
d_model	p_drop	weight_decay	vocab_size	BPC	Stopped Early
128	0.1	0	2000	1.305	No
256	0.1	0	2000	1.329	Yes
256	0.1	0.1	2000	1.268	Yes
256	0.2	0.1	2000	1.230	Yes
256	0.2	0	2000	1.274	Yes
256	0.3	0.2	2000	1.217	No
256	0.5	0.3	2000	1.321	No
256	0	0	8000	1.820	Yes
256	0.2	0.1	8000	1.247	Yes
256	0.3	0.2	8000	1.218	No
256	0.5	0.3	8000	1.268	No
256	0.1	0	14000	1.361	Yes
256	0.3	0.2	14000	1.249	No
512	0.1	0	2000	1.376	Yes
512	0.1	0.1	2000	1.377	Yes
512	0.3	0	2000	1.280	Yes
512	0.3	0.2	2000	1.244	Yes
512	0.4	0.2	2000	1.239	Yes
512	0.5	0.3	2000	1.226	No
512	0.5	0.5	2000	1.240	No
800	0.5	0.3	2000	1.268	Yes
800	0.6	0.5	2000	1.290	Yes
800	0.4	0.4	10000	1.448	Yes
800	0.5	0.5	10000	1.361	Yes

(b) Hyper-parameter tuning for isiZulu model trained only on data from all Nguni languages.

Hyper-Parameters				Metrics	
d_model	p_drop	weight_decay	vocab_size	BPC	Stopped Early
400	0.1	0.1	2000	1.198	Yes
480	0.3	0.2	2000	1.167	No
512	0	0	500	1.332	Yes
512	0.1	0	500	1.230	No
512	0.1	0	2000	1.229	Yes
512	0.1	0.1	2000	1.202	Yes
512	0.2	0.2	2000	1.169	Yes
512	0.3	0.2	2000	1.154	No
512	0.5	0.4	2000	1.251	No

(c) Hyper-parameter tuning for isiZulu model trained only on all non-European South African language data.

Hyper-Parameters				Metrics	
d_model	p_drop	weight_decay	vocab_size	BPC	Stopped Early
512	0.1	0.1	2000	1.209	No
512	0.1	0.1	8000	1.188	No
512	0.2	0.1	8000	1.163	No
512	0.3	0.2	8000	1.192	No
512	0.3	0.2	8000	1.202	No
968	0.3	0.2	8000	1.143	No