Review of Language Modelling Techniques for use in Low Resource Settings

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

ABSTRACT

Recent work in natural language processing (NLP) has shown great advances in synthetic benchmark scores when given sufficient training data. This paper aims to provide an overview of the current state of language modelling techniques - specifically examining N-gram, Long Short Term Memory (LSTM) and Transformer variants - and their potential effectiveness on low resource South African languages.

CCS CONCEPTS

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

KEYWORDS

language modelling, neural networks, low-resource languages, LSTM, N-Gram, Transformer

1 INTRODUCTION

Statistical language modelling is an important part of many natural language processing tasks such as speech recognition, translation, auto correct and summarisation amongst others [7, 24]. 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.

This paper examines the three main types of language models used: N-grams [3], Long Short-Term Memory models (LSTM) [13] and Transformers [28]. We hypothesise that the recent state of the art architectures for LSTM and Transformer models will continue to show good performance on low resource languages, specifically South African Nguni languages such as Zulu and isiXhosa.

2 LANGUAGE MODELLING

A statistical language model can be defined as the probability distribution of a set of strings over a given context[3, 24], or as a model capturing the probability of a string S occurring in a given context C. Defining this context and how it is derived is the key differentiating factor between the three types of models examined. Properly trained models are able to capture more complex structures such as grammatical rules based only on this statistical distribution.

3 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.[15] 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.

Perplexity

One way to measure the performance of a model intrinsically is by measuring the probabilities assigned to a sequence on which the model has not been trained, relative to the true probabilities found in the test set. In practice, perplexity is used which measures the inverse probability assigned to the test set, normalised for the number of words in the test set. [15] 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.

Cross Entropy

For a variable x_i in a given distribution X with probability $P(x_i)$, the entropy of x_i is calculated as in Jurafsky and Martin[15]:

$$H(X) = -\sum_{x_i \epsilon X} p(x_i) \log_2 p(x_i)$$

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 [15]:

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

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. [24]

4 MODEL ARCHITECTURES

N-Gram

N-gram models are the simplest of the three models discussed. N-gram models 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[15]. 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. Following this, n-gram models are based on the frequency of a word S in a sequence, relative to the frequency of all other sequences of the same length in the training corpus.

N-gram model complexity is related to the number of potential sequences and 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. Conversely, it is initially simple to scale the models to take advantage of different sizes of dataset. A number of techniques have been proposed to allow the use of a combination of large n-grams, with smaller n-grams used to provide additional information for less frequently occurring sequences. [3]

N-gram models typically have trouble capturing long term dependencies where there are large gaps between the relevant parts of the sequence. This is due to the exponential increase in the number of possible n-grams, and therefore model size, as you increase the "window" or "context" size of the n-gram model. This results in n-grams being unable to model long term dependencies as the "context" available extends back only N words in the sequence. [5, 27]

A limitation of n-gram models which rely purely on the absolute frequency of sequences in the training set is their inability to handle unseen n-grams[1]. Given some unseen sequence $W_1 ... W_n$, a model would output a probability of 0 for the sequence. This causes issues both in synthetic tests such as perplexity as well as real world applications. For example, if such a model were used to assist in a speech-to-text application, regardless of the clarity of the spoken sequence it would never select the correct results.[3] To alleviate this issue the concept of *smoothing* was devised whereby the probability estimates of a model are adjusted in an attempt to improve the accuracy of the model. Smoothing algorithms tend to raise low (or zero) probability sequences whilst lowering the probability for very high likelihood sequences. Intuitively this *smooths* the distribution of all probabilities in a model.

A popular variant on the standard n-gram model, Kneser Ney smoothing [17], and its modified version, initially proposed in Chen et al. [3] and improved[2] further by Heathfield et al. [12] provides an efficient smoothing method for building n-gram models. Kneser-Ney smoothing is a form of backoff smoothing in which higher order [3] n-grams are combined with lower order models when a given sequence or n-gram has a count of zero.

In addition to Kneser-Ney smoothing, Katz smoothing and Jelinek-Mercer smoothing present the next best performing models with Jelinek-Mercer outperforming Katz in small datasets although Kneser-Ney performed best overall in situations where lower order n-grams made up the majority of the model (which is more often the case when dataset sizes are small). We hypothesise that the performance of the n-gram models in these smaller datasets will translate well to tests on low resource languages which are also characterised by similarly small datasets.

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, 14]. 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]

[1]Here "n-gram" refers to a sequence of token of length n

[2]Heafield et als. improvements are in the implementation efficiency, rather than an improvement in model performance. However their approach enables large, more scalable models to be used in turn leading to better performing models on certain tasks.

[3]The order of an n-gram model refers to the size, N, of the context window

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 [13]. 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)

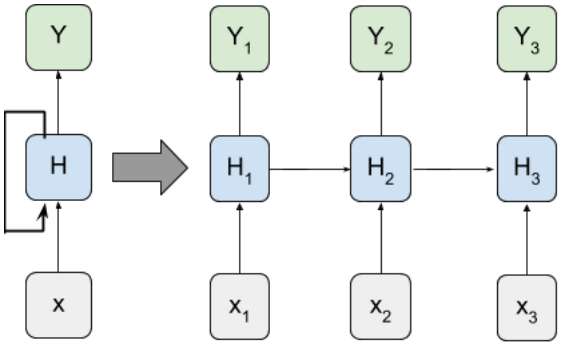

Figure 1: RNNs can be "unrolled" better showing the flow of information between timesteps

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 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: [9]

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

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) [28]

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 back-propagation through time (BPTT) algorithm allows for each hidden layer to contribute to the loss at the previous timestep. This repeated multiplication gradually pushes the gradients towards zero as sequence lengths increase. [16, 30] 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.

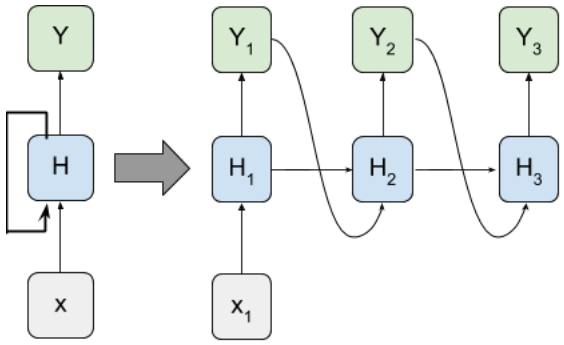

Figure 2: A simple RNN network can be "unrolled" as a sequence of RNN cells, the output of each cell being fed as input to the next

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 model with input and state size m and n respectively can be described more formally in Melis et al. as: [19]

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

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

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. This becomes particularly apparent in large models due to memory constraints. Furthermore, without proper regularisation's LSTMs often suffer from overfitting the training data. [21]

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* [26] are not successful when applied between timesteps on the recurrent connection as it inhibits the models ability to retain long term dependencies. [21, 31] Merity et al. introduced the use of a weight-dropped LSTM using DropConnect[29] 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 [21].

Merity et al.[21] examined a range of regularisation techniques, of which variational dropout[6] 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.

AWD-LSTM

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, the AWD-LSTM.[21] In addition to the specified regularisation strategies, it includes the use of a neural cache [10], whereby the previous hidden states are stored and the probability distributions produced by the model and cache are combined to give a final prediction. The neural cache is used as it was shown to given an improvement on perplexity scores of 6-11 points in the PennTreebank(PTB) and Wikitext-2(WT2) datasets respectively. Moreover the hyperparameters can be tuned on the validation set after having trained the model making it a relatively inexpensive technique.[21]

Morgrifier

Morgrifier LSTM networks represent the current state of the art RNN models and achieve state of the art performance on certain datasets. Melis et al. [19] proposed an extension of the standard LSTM in which the input is gated conditioned on the output of the previous step resulting in an contextualised representation of the input. This is achieved by gating the input X^0 on the output of the previous step h^0_{prev}, followed by gating the output of the previous step (h^0_{prev}) with the gated X value (X^1). This can be repeated more than once before feeding the resultant X^n and h^{n-1}_{prev} into the LSTM. Setting the number of round, n, to zero results in a standard LSTM. As such the Morgrifier can be thought of as a preprocessing step allowing the possibility of pairing it with other LSTM variants which may lead to further improvements.

The model was tested on the PTB and WT2 datasets and shown not only to outperform the standard LSTM models by 2.8-4.3 perplexity points [19], they also present a new state of the art performance over the previous FRAGE [8] and AWD-LSTM [21] models in both word level and character level tests.

Furthermore in the larger Enwik8 character level dataset, although unable to outperform the results achieved using Transformer-XL[4], it was able to close the gap in cases where model size was kept relatively small at 40-50M parameters. Additional tests on other language modelling tasks showed that the model was capable of extending to non-English datasets.

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 larger word-level datasets or in character level models was not found.

Transformer

Transformers are a relatively new form of language model which rely solely on attention mechanisms, removing the need for recurrence in the networks [28]. 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.

Standard transformer models are held back by the reliance on a fixed length context which they are capable of encoding. Transformer-XL [4] provides an approach to enabling variable length input sequences to be processed and dependencies longer than the length of the context to be captured. Additionally performance is greatly improved as context fragmentation (the splitting of sentences or contexts due to fixed size context window) is reduced. This is shown by testing the model on the One Billion Word dataset which does not benefit from long term dependencies, isolating the improvements to the reduced context fragmentation. Specifically, reducing context fragmentation results in an improvement in perplexity from 27.1 to 25.2 points. [4] Transformer-XL introduces recurrence in the form of reusing the hidden states obtained in previous segments (more than one segment may be cached, depending on the GPU memory available), the hidden states thus act as "memory" resulting in the ability to model much longer dependencies. Consequently the absolute positional encoding of standard transformers is replaced with relative positional encoding, further helping the model to generalise to sequence lengths longer than those seen during training.

Due to their ability to model long range dependencies, Radford et al. [23] 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-2, is an order of magnitude larger than the previous GPT model produced by OpenAI.

5 DISCUSSION/COMPARISON OF METHODS

Gandhe et al. [7] showed that in spite of the current state of the art models all being built from either Transformer or RNN architectures, n-gram models have the potential to outperform RNN models when small datasets (less than 100k) are used. However this could not take into account the improvements offered by LSTM networks and other variants on the RNN model.

We take particular interest in the results on both the PTB and WT2 results of the discussed models as these are the smallest of the commonly examined datasets and as such are most likely to approximate the performance in low resource languages where datasets are similarly restricted in size.

Chen et al. [3] showed in a comparison of different n-gram model types, including both backoff and interpolated smoothing, that modified Kneser-Ney smoothing consistently outperforms all other tested models across multiple datasets.

Character level language models(LMs) are useful in reducing the vocabulary size of a model whilst increasing the number of tokens that must be generated whilst a word level LM encodes more information into each token but is computationally harder to train due to the increased vocabulary size. Conversely, character level LMs are slower to process than word level LMs and are more susceptible to the vanishing gradients problem as many more tokens are needed to provide context. [20]

Byte Pair Encoding (BPE) [25] represents a useful middle-ground between character and word level models allowing the vocabulary size to be tuned as an additional hyperparameter in the model. [22, 23] This is achieved by splitting word level tokens into smaller subword tokens which are common amongst many different words (although in reality these sub words are built up by merging common pairs of tokens starting from a character level vocabulary). Additionally as the set of subwords used to build up the model vocabulary contains all the characters in the training set as tokens, any novel word in the test set is still able to be explained by the model. This "subword" technique is particularly useful in languages where compound words are common, such as German, where a concept which would in English be conveyed through multiple words is written as a single compound word. [25]

In addition to the models discussed above, it has been shown that in certain cases, provided enough compute resources, a properly optimised and well trained more standard model can produce comparable results and even out perform some more complex models.[18] This is due to model performance depending not only on the model architecture and data available during training but also the tuning of the hyperparameters. The growth in computing power available today makes it affordable to control for hyperparameter tuning through the use of black box optimisers in spite of the large computational costs involved.

Transformer models have been shown to be able to model complex and long term dependencies making them particularly well suited for language modelling tasks. However due to the performance seen when trained on large datasets, little data is available to reliably predict performance in low resource settings, however on the PTB dataset transformer-xl set a new state of the art. In large datasets Transformer models readily outperform previous state of the art models. This suggests transformer models may generalise well even on smaller datasets, although this is for specifically tuned models. The more generalised GPT-2 is still prohibitively expensive to train at this stage.

Based on these results we hypothesise that the Morgrifier or AWD-LSTM variants should produce acceptable results on the low resource languages to be examined. This is backed up by their strong performance on a wide range of datasets with particular emphasis on the PTB and WT2 datasets which are relatively small compared to the more popular, large scale datasets that transformers excel on. Whilst not discounting

n-grams for datasets which are particularly small, previous results suggest that RNNs ability to model longer term dependencies will produce better results. Transformers may also produce good results based on the strong performance on the PTB dataset set by Transformer-XL.

6 CONCLUSIONS

Due to the use of large scale datasets in language modelling research many of the new state of the art models have been trained and evaluated on datasets orders of magnitude larger than those considered to be available in "low resource" languages. However the results from testing models against the PTB and WT2 datasets could reasonably approximate the results we hypothesise to find when training and evaluating on South African Nguni languages for which large high quality datasets are not widely available.

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] 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] Stanley F Chen and Joshua Goodman. 1999. *An empirical study of smoothing techniques for language modeling.* Technical Report. 359–394 pages. http://www.idealibrary.comon

[4] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. (jan 2019), 2978–2988. arXiv:1901.02860 http://arxiv.org/abs/1901.02860

[5] 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

[6] 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.

[7] 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.

[8] Chengyue Gong, Di He, Xu Tun, Tao Qin, Liwei Wang, and Tie-Yan Lie. 2018. Frage: frequency-agnostic word representation. In *Advances in Neural Information Processing Systems.* 1334–1345.

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

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

[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] Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H Clark, and Philipp Koehn. 2013. Scalable Modified Kneser-Ney Language Model Estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers).* Association for Computational Linguistics, Sofia, Bulgaria, 690–696. https://www.aclweb.org/anthology/P13-2121

[13] 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

[14] 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

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

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

[17] R. Kneser and H. Ney. 1995. Improved backing-off for M-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, Vol. 1. 181–184 vol.1.

[18] Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the State of the Art of Evaluation in Neural Language Models. In *International Conference on Learning Representations.* arXiv:1707.05589v2

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

[20] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. An Analysis of Neural Language Modeling at Multiple Scales. (mar 2018). arXiv:1803.08240 http://arxiv.org/abs/1803.08240

[21] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings.* International Conference on Learning Representations, ICLR. arXiv:1708.02182

[22] Sabrina J Mielke and Jason Eisner. [n.d.]. *Spell Once, Summon Anywhere: A Two-Level Open-Vocabulary Language Model.* Technical Report. John Hopkins University, Baltimore, MD, USA. arXiv:1804.08205v4 www.aaai.org

[23] 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

[24] Alessandro Scarcella. 2018. *Recurrent neural network language models in the context of under-resourced South African languages.* Technical Report. University of Cape Town, Cape Town. https://open.uct.ac.za/handle/11427/29431

[25] 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. https://doi.org/10.18653/v1/P16-1162

[26] 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.

[27] 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

[28] 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

[29] 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.

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

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