Canonical Morphological Segmentation for Nguni Languages using Hard Attention

**Simbarashe Mawere**

**Supervised by Jan Buys**

Department of Computer Science

University of Cape Town, South Africa

MWRSIM003 at myuct dot ac dot za

# Abstract

The task of Morphological Segmentation is one explored in Linguistics concerning the decomposition of words into their composite morphemes. This project focuses on the task of Canonical Morphological Segmentation of Nguni Languages to extract underlying morphemes from words utilising sequence-to-sequence models which used both soft-attention and hard-attention machine learning. The first implementation includes a hard-attention Neural Transducer (HNT) and a soft-attention Transformer (ST) while the second implementation is a soft-attention Transformer baseline (BaselineT). The HNT and the ST outperform the baseline with the HNT outperforming both transformers with an optimised average F1 score of 75.11% in all four languages. The results also show that modifying the parameters of the Transformer can result in the improvement of results, illustrated by the optimised ST performing 1.93% better than the baseline transformer. The expectation is that the explored models will be better optimised in the future through an iterative experimental process for use in the natural language processing (NLP) of Nguni languages and other low-resource or agglutinative languages.

Keywords: natural language processing, morphological segmentation, canonical segmentation, transformer, attention, hard attention

# Introduction

In the computational linguistics field of NLP, segmentation is an important task that involves splitting structures such as words, phrases, or sentences into composite substructures [[3]](#_Bibliography). Morphological segmentation refers to the decomposition of words into morphemes which are the smallest meaning-bearing units of language, while canonical morphological segmentation is segmentation that involves the extraction of underlying morphemes which are not visible on the surface. The purpose of the extraction of morphemes is to preserve languages and develop of NLP tools for use by speakers and learners of the languages.

This morphological segmentation is useful in helping linguists in the preservation of these languages [[9]](#_Bibliography) by automating tasks that would otherwise have been performed manually by human beings, costing time and human resources. These languages also contain rich information in the form of morphemes and analysis of these can be used to explore further NLP tasks such as web-crawling using search engines, keyword spotting and more acute translation of these languages [[5, 12, 15]](#_Bibliography). The morphemes would be used to extract the meaning of the word and find translations that would have been difficult to decipher without the segmentation. Furthermore, models for morphological segmentation may be used in studying Linguistics [[14]](#_Bibliography).

This work and exploration seek to improve the accuracy of the task by utilising a mechanism that has not yet been explored for the stated task, especially for Nguni Languages, in the existing literature. The task is exemplified by the word “zobomi” from the isiZulu dataset.

This is illustrated in Table 1 below:

|  |  |  |
| --- | --- | --- |
| Word | Surface segmentation | Canonical segmentation |
| zobomi | zo-bo-mi | za-u-bu-omi |

Table : An example of the difference between surface and canonical morphological segmentation

In this project, the aim is to explore models for undertaking the abovementioned task for Nguni languages using character-level sequence-to-sequence models. Afterwards, there will be a comparison of the performance of hard attention to soft attention as implemented by Moeng et al. [[11]](#_Bibliography) with the expectation to improve performance. A hard attention architecture [[19]](#_Bibliography) will be trained to perform canonical morphological segmentation and its performance will be evaluated on the same test data using the same evaluation metrics. Furthermore, the project also aims to improve the performance of the Transformer with regular soft attention as compared to the baseline by substantial percentage points. This will be attempted by altering key hyperparameters on an existing implementation of the Transformer [[19]](#_Bibliography).

The models explored in the experiments are used for the analogous grapheme-to-phoneme (g2p) sequencing, which takes input as a sequence of space-separated characters and outputs a similar sequence with the phoneme version of the grapheme. Due to the similarity of the input taken by this model to the data from Moeng et al. [[19]](#_Bibliography), the model architecture is adapted but the focus is on training the models for the task of morphological canonical segmentation. Since canonical segmentation extracts underlying morphemes that may not be present on the surface, the input and output sequences probably differ in length.

The experiments’ results showed that the HNT and the ST models were outperforming the baseline Transformer achieving average F1 scores of 2.56% and 1.69% respectively, across all four languages when evaluated using the metrics and testing data.

# Literature Review

There have been previous attempts at applying transformers and hard attention to the task of morphological segmentation. Mager et al. [[7]](#_Bibliography) tackle canonical morphological segmentation for the low-resource languages Tepehua and Populuca for language preservation. A Neural Transducer with Imitation Learning using hard monotonic attention was trained using ADADELTA [[23]](#_Bibliography) optimisation as opposed to a soft attention Pointer-Generator network. The transducer was trained for 30 epochs and achieved F1 scores of 44.0% and 54.7% for the two languages respectively, using 900 data points for each language. Both languages explored in the study are low-resource polysynthetic languages. Like agglutinative languages, they carry meaning through different morphological combinations. In their experiments, the hard-attention Neutral Transducer outperformed the F1 performance of the soft-attention Pointer-Generator Network.

The baseline performance for this research was acquired from the study of Moeng et al [[10]](#_Bibliography) which implemented a Transformer architecture using soft attention for the exact task of canonical morphological segmentation in Nguni Languages. They compared it with other character-level sequence-to-sequence models such as the Long Short-Term Memory (LSTM), the bidirectional LSTM (BiLSTM) and rule-based models. The trained Transformer outperformed all the other models with more 7.13% F1 score difference between it and the next-best result. Like this study, their model made use of ADAM optimisation [[6]](#_Bibliography) but with parameters: a dropout rate of 0.3, a learning rate of 0.0005 and a hidden dimension of 256. There was a significant step up in the performance of the Transformer in comparison to the other models which makes it a viable model for exploration.

The HNT and ST model architectures were acquired from the neural transducer [[19]](#_Bibliography) codebase on GitHub and were both used for transduction tasks at the character level. One of the tasks explored in the study is grapheme-to-morpheme segmentation which is comparable to canonical segmentation. Wu et al. and Wu and Cotterell [[17, 20]](#_Bibliography) provided the basis for hard attention with implementations of hard attention model architecture. For the soft attention Transformer architecture, the codebase was again the neural-transducer GitHub but used another study by Wu et al [[18]](#_Bibliography). The hard attention neural transducer outperformed the soft attention neural transducer in this for both its large and small configurations. The small neural transducer was configured with an embedding size of 100, 200 encoder layers and 200 decoder layers while the large configuration had the stated hyperparameters doubled.

# Methodology

## Data Processing

The data as aforementioned was acquired from the NCHLT Speech Corpus [2] in the form given in Figure 1. Of the three character-level sequence tasks explored by Wu et al. [18, 20], the g2p conversion task was most similar in both sequence length and form. The models took the input in the format shown in Table 2 below meaning that the input from the speech corpus had to be moulded into an identical format to match the expectation.



Figure : Data as presented in the NCHLT Speech Corpus

|  |  |
| --- | --- |
| Input sequence | Output sequence |
| n g o m t h o m b o | n g a - u - m u - t h o m b o |

Table : An example of the formatting for the input and output sequences

The input files were formatted to have the input and expected output tab-separated while the individual tokens were space-separated. The individual morphemes in the output sequence are demarcated by the hyphen “-“ character. The data loader for the model loaded all the individual unique tokens for use in the model. The data for each language were adapted as used by the MORPH\_SEGMENT code base [10] and the splits that were used were identical. This was to minimise the variations between the baseline and the experiments. The training, development, and testing dataset sizes for the four languages are shown in the table below in Table 3:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Language | Training | | Development | | Testing |
|  | Total | Unique | Total | Unique |  |
| isiZulu | 17778 | 9410 | 1777 | 9410 | 3298 |
| isiXhosa | 16879 | 9078 | 1688 | 1461 | 3004 |
| isiNdebele | 12929 | 7225 | 1292 | 1123 | 2553 |
| siSwati | 10791 | 3783 | 1080 | 807 | 1347 |

Table : The dataset distributions for all four Nguni languages

The data are split into an approximately 78:8:14 split for training, development and testing respectively with the siSwati data set being the outlier with an approximate split of 82:8:10. The development dataset was a subset of the training dataset which is speculated to be due to the lack substantial datasets. Furthermore, in both the training and development datasets there were repetitions of words within the sets, which is also shown in Table 3 where the number of unique words is significantly lower than the number of words present.

## Models

In prediction with a model using any type of attention there is a context vector which represents a character in the input sequence at a time *j*.

Soft attention is a mechanism that allows for all individual hidden states of the input characters to be input into the model to be fed into the decoder to produce an output sequence of characters. With being the decoder hidden state for the *i*th output character. would be the encoder’s hidden state at step *j* in the encoding process. To calculate the context vector, , one would need to calculate the relevance of the previous decoder hidden state and all the encoder hidden states. This relevance score is the dot-product attention, , illustrating how similar the input encoder states are to the concerned decoder hidden state given by:

|  |  |
| --- | --- |
|  | (1) |

The scores are normalized using a softmax function to make them comparable positive values which add up to one. The softmax, , for a given decoder step and j encoder steps is given as:

|  |  |
| --- | --- |
|  | (2) |

This is calculated for each value of j.

The difference between hard attention and soft attention comes in when the context vector, needs to be calculated. is taken to represent the location variable where the model decides to focus its attention on when generating the *i*th output character. Xu et al. [21] report that when using soft attention, the expectation of the context vector, is calculated directly from all *j* encoder states, containing the annotation vectors, and the softmax score for the step as in Equation 3:

|  |  |
| --- | --- |
|  | (3) |

From this, a deterministic attention model is formulated for computing soft attention weighted annotation vectors. This is the soft attention used in BaselineT and ST.

Hard attention, alternatively, uses a stochastic attention model [1]. For this, the location variable for each input character *i* at the *j*th step in the encoder, , is required and is set to 1. This location variable is useful in extract the features in the character sequence. The locations of attention, represented by the variables are treated as intermediate variables to make a distribution parametrized by the softmax, as follows:

|  |  |
| --- | --- |
|  | (4) |

A context vector is calculated for each character input as:

|  |  |
| --- | --- |
|  | (5) |

The equation illustrates that when using hard attention, the context vector is calculated at each step as a weighted average of the encoder annotation vectors and the location variable at that point. Comparing equations (3) and (5), the difference between hard attention and soft attention becomes apparent. Hard attention varies intrinsically to soft attention in that the context vector for each output character is aligned with a corresponding input character while for soft attention an overall context vector for each output character is calculated from a weighted average of all the input hidden states [21].

## Preliminary Experimentation

All the experiments were run on the National Integrated Cyberinfrastructure System’s Centre for High-Performance Computing (CHPC) cluster. Due to the utilised models being complex and involving multiple hyperparameters, preliminary experiments were conducted using the scripts provided for each model using the “interactive session” mode of the CHPC. This mainly consisted of running the models in their default states to gauge performance and analyse the input and output sequences. It was due to this preliminary experimentation and model training that the g2p conversion task models were chosen over the other two, and the format for the data was acquired.

## Experimental Setup

Both the HNT and the ST data were formatted in the format of Table 2 since the data loader was the same for the g2p conversion task, adapted for canonical segmentation. The models were implemented by adapting the code base of Wu et al. [[17, 18]](#_Bibliography) which was free for use on GitHub. As shown in the sample output sequence, the generated output contains the hyphen (“-“) character as it is recognised as the morpheme demarcation boundary for the individual morphemes in the output. The empirical part of the project was to determine the better model from the present options based on the F1 scores achieved in development.

There are three architectures considered in the experiments: the ST and a hard attention neural transducer with a large (HNT Large) and small (HNT Small) configuration. The main differences between the large and small configurations are that HNT Large has two source encoder layers while the HNT Small is configured with only one. Furthermore, the larger model has gradient clipping with a max norm of 5 compared to the small which is not clipped.

For each of the models, there was a set of hyperparameters which were explored. These were mainly the dropout rate, embedding size and encoder/decoder size. The hyperparameters are tabulated below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter | HNT Small | HNT Large | ST | Values trialled |
| Dropout | 0.2 | 0.3 | 0.2 | {0.2, 0.3, 0.4, 0.5, 0.6} |
| Emb. size | 100 | 300 | 300 | {100, 200, 300, 400, 500} |
| Enc. size | 200 | 400 | 1024 | {200, 400, 600, 800, 1000, 1024) |

Table : The hyperparameters use in tuning the hard attention models

The configuration of used 10 CPUs and 1 GPU, running with a maximum time of 6 hours per trial.

## Optimisation/Hyperparameter Tuning

All the models were trained with the Adam optimiser [[6]](#_Bibliography), and it was found that an initial learning rate of 0.001 was the most effective. Both HNT models were trained for at most 50 epochs while the ST was trained for a maximum of 20000 training steps which translated to between around 400 and 800 epochs depending on the vocabulary size of the language. The training was performed using an early-stopping mechanism which stopped the training when the loss stopped decreasing and started increasing again. Loss is defined as an error in prediction by a neural network and the adapted codebase utilises the negative log-likelihood (NLL) loss function to calculate it [[13, 22]](#_Bibliography). The hyperparameters are tuned using the range of values shown in the last column of Table 4.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **dropout** | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 |
| **ST** | **99.46** | 99.36 | 98.66 | 96.55 | 91.69 | 84.09 |
| **HNT Large** | **99.12** | 99.12 | 99.10 | 98.98 | 98.75 | 98.17 |
| **HNT Small** | **99.13** | 99.03 | 98.88 | 98.71 | 98.35 | 96.03 |

Table .1: Average development performance for the models at various dropout rates

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Emb. Size** | 100 | 200 | 300 | 400 | 500 |
| **ST** | 99.38 | 99.44 | 99.47 | **99.52** | 99.31 |
| **HNT Large** | 99.13 | 99.12 | **99.19** | 99.15 | 99.06 |
| **HNT Small** | 99.03 | 99.14 | 99.13 | **99.16** | 99.16 |

Table 5.2: Average development performance for the models at various embedding dimension sizes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Enc. Size** | 200 | 400 | 600 | 800 | 1000 |
| **HNT Large** | 99.08 | 99.14 | **99.16** | 99.05 | 98.96 |
| **HNT Small** | 99.08 | **99.12** | 99.09 | 99.01 | 98.96 |

Table 5.3: Average development performance for the models at various encoder/decoder sizes

The dropout values that were experimented with are shown in Table 5.1 with all the models performing their best at lower values of dropout rate. Dropout is a mechanism that drops or ignores units at random during training to prevent over-fitting of the model to the test and development data [16]. The dropout rate is what proportion of the units are dropped and for experimentation, it was limited to a minimum of 0.2 to avoid overfitting the models to the training data. The models were then retrained with different embedding dimensions at the optimal dropout rate of 0.2. The best development values were selected with the HNT Small and ST peaking at 400 while the HNT Large was at 0.3 as shown in Table 5.2 Finally, the encoder/decoder dimensions were altered with the optimal dropout rates and embedding dimensions and both hard attention models peaked at 400 in Table 5.3. The ST was already optimised at a very large value of the encoder/decoder dimension of 1024 which was not in the search range. Furthermore, it was discovered from the two hard attention models that by changing the value of the hyperparameter there was no increase in performance from the default values of 400 and 600 for the small and large configurations respectively.

## Evaluation

For evaluation, the least loss model is selected by the metrics from the Wu et al. neural transducer [[19]](#_Bibliography) and the Wu et al transformer [[18, 20]](#_Bibliography), included with the training of the model. However, for the tuning and optimisation of the hyperparameters, the F1 score is used since the results are to be compared to those from Moeng et al [[11]](#_Bibliography). F1 is a metric for measuring how precise and good at recall a model is at predicting [[4]](#_Bibliography) as given by the equation:

Precision is the proportion of correct identifications; in our case, morphemes in the output were in the expected output as a proportion of all morphemes in the predicted output. Recall, on the other hand, is the proportion of the actual positives which were identified correctly [[4]](#_Bibliography). The version of each model with the highest F1 score was selected and the results were tabulated in Tables 6 and 7 below.

# Results

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Ndebele** | | | **Swati** | | |
|  | P | R | F1 | P | R | F1 |
| **ST** | 69.227 | **75.247** | **72.112** | 72.995 | 75.996 | 74.465 |
| **BaselineT** | **73.14** | 66.67 | 69.76 | **76.07** | 72.96 | 74.48 |
| **HNT Large** | 69.97 | 74.06 | 71.96 | 72.55 | **76.51** | 74.48 |
| **HNT Small** | 69.98 | 73.89 | 71.88 | 72.97 | 76.10 | **74.50** |
|  |  |  |  |  |  |  |
|  | **Xhosa** | | | **Zulu** | | |
|  | P | R | F1 | P | R | F1 |
| **ST** | 71.504 | 78.807 | 74.978 | 72.63 | 78.347 | 75.38 |
| **BaselineT** | **75.76** | 68.36 | 71.87 | **77.34** | 71.04 | 74.06 |
| **HNT Large** | 71.84 | 79.37 | 75.42 | 73.94 | 81.77 | 77.66 |
| **HNT Small** | 72.55 | **79.60** | **75.91** | 74.62 | **81.97** | **78.13** |

Table : Results from experiments showing the best performance per model per language

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Average** | | |
|  | P | R | F1 |
| **ST** | 71.59 | 77.10 | 74.23 |
| **BaselineT** | **75.58** | 69.76 | 72.54 |
| **HNT Large** | 72.08 | **77.93** | 74.88 |
| **HNT Small** | 72.53 | 77.89 | **75.11** |

Table :Results from experiments showing the average best performance of each model across all four languages

## Altering the parameters of the Soft Attention Transformer

Tables 5 and 6 show a difference in the best performances of the baseline Transformer implemented by Moeng et al. (BaselineT) [[10]](#_Bibliography) and the Transformer implemented by Wu et al [[19]](#_Bibliography). with the latter being the best model in terms of F1 score. The exception for this is in Swati in which the F1 score for BaselineT was 0.015% higher than for ST. This difference is attributed to BaselineT having its best performance for Swati which Moeng et al. report is due to the language having the shortest word length [[11]](#_Bibliography). This difference is, however, minuscule enough to be negligible. The Swati dataset is also the dataset in which all four models perform on par with each other and this can be attributed to the limited training data for Swati as shown by Table 3, in which it has a higher percentage of repeated words than other languages. This in turn makes the BaselineT perform against the trend with significantly higher results than expected.

Overall, however, there is a 1.69% average improvement by ST from the BaselineT as shown in Table 7. For ST, the languages with the larger data sets perform better on the test data but for BaselineT, the data do not show a trend linking performance to the dataset size. Zulu shows the most improvement which could be attributed to it being the language with the largest training dataset. On average, the altering of parameters in the transformer improves the results. The noted changes in parameters are given in the table below:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Dropout | Learning rate | Hidden dim. |
| BaselineT | 0.3 | 0.005 | 256 |
| ST | 0.2 | 0.001 | 300 |

Table : The different hyperparameters for BaselineT and ST models

The two models, while both soft attention transformers are different in the hyperparameters they use and therefore accomplish different results. It is, however, worth noting that these were not the only parameters that were different between the two models but rather the ones which the ST was optimised for in this project. It is illustrated that altering the parameters of the soft-attention Transformer can better the performance of the model on the task of canonically segmenting Nguni languages.

## Comparison of Hard Attention to Soft Attention

By comparing the hard attention, HNT results to the soft attention, ST and BaselineT, in Table 5, the HNT models outperform the soft-attention models for all languages except Ndebele in which the ST has the best F1 score. The small configuration HNT has the highest F1 of all the models on average in Table 6, exceeding the best soft-attention performance by 0.88%. The greatest difference between the small HNT and the ST is observed on the Zulu dataset at 1.28% while the least improvement is on the Ndebele dataset in which the ST had the better performance by 0.16%. While the results are close for the HNT and the ST, there is a clear difference between the HNT and the BaselineT with the average score over the four languages being a 2.56% increase from the small configuration HNT to the BaselineT. The small configuration proves to be the better-suited trained model for canonical segmentation than the large configuration even though they are both better the soft attention.

The difference between all three models from the neural-transducer codebase seems to be very minimal in comparison to their difference with the BaselineT. Following this discovery, the improvement from the BaselineT to the HNT cannot be concluded to be a result of using hard attention over soft attention. Rather the results suggest that the configurations of neural-transducer models were better suited for the canonical morphological segmentation task than the one in MORPH\_SEGMENT.

The results also show that the two hard attention implementations have very little difference in their performance. This can be attributed to an assertion that the dropout rate, embedding dimension and encoder/decoder size were key defining features for “large” and “small”. Therefore, altering those hyperparameters fundamentally made the two configurations nearly identical except for the gradient clipping and the source encoder layer, which seem to have had minimal effect on the results. The maximum difference between the large and small configurations is 0.49% which is small enough to be discarded in comparison to the overall accuracies achieved of over 70%.

# Conclusions and Future extensions of work

Three different models were trained and developed for the task canonical segmentation for the four Nguni languages: Ndebele, Swati, Xhosa, and Zulu. The models were implemented using both hard and soft attention for comparison. It was discovered that hard attention models performed better than soft attention models. The difference in performance was just 0.88% between the best soft attention and hard attention models. This difference, nevertheless, was not taken to be significant enough to conclude that the hard attention caused an increase in the performance of the model. Rather, the overall increase in performance from the baseline was attributed to the neural-transducer codebase being better optimised for the task.

The results also showed that altering fundamental optimisation parameters in a model, especially a Transformer, could lead to a substantial difference in the performance of one configuration compared to another. The modified Transformer achieved an average of 74.23% which was 1.69% higher than that for the baseline Transformer. This illustrated that there is a possibility of increasing performance if the right hyperparameters are altered.

The performance of the models was shown to vary between the four languages, proving that a model can perform well for one language structure but not as well for another. This was illustrated by the difference in the performance of the Ndebele dataset which performs drastically differently from the other datasets. It has better performance using the modified Transformer than it has using the hard attention Neural Transducer configurations as opposed to the three other languages which perform better using Neural Transducer. The task of canonical morphological segmentation can be better improved if, for future experiments, the optimisation is carried out per language not using average values. This would help to improve the performance since each language, while similar in agglutination, has differences which alter the performance of the model such as average word length and morphemes per word.

Furthermore, the experiments in the project were stunted by a lack of data and as such, the models did not have access to as much unique training data as would be ideal. Performance can be improved in the future by finding larger datasets of canonical segmentation. With average performance peaking at 75.11%, there is still room for improvement to make the models better at the task and further exploration can achieve that.

## Deliverables

The optimised models and instructions on how to run them are uploaded to the GitHub MORPH\_SEGMENT2 repository [8] along with the scripts. All the code developed in this project is documented with instructions for use.

# Bibliography

[1] Adaloglou, N. 2020. How Attention works in Deep Learning: understanding the attention mechanism in sequence models: 2020. *https://theaisummer.com/attention/*. Accessed: 2022-10-06.

[2] Barnard, E., Davel, M.H., van Heerden, C.J., de Wet, F. and Badenhorst, J. 2014. The NCHLT speech corpus of the South African languages. *SLTU* (2014).

[3] Cotterell, R., Kumar, A. and Schütze, H. 2016. Morphological Segmentation Inside-Out. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (Austin, Texas, Nov. 2016), 2325–2330.

[4] Google. 2022. Classification: Precision and Recall: 2022. *https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall*. Accessed: 2022-08-27.

[5] Grönroos, S.-A., Virpioja, S. and Kurimo, M. 2018. Cognate-aware morphological segmentation for multilingual neural translation. *Proceedings of the Third Conference on Machine Translation: Shared Task Papers* (Belgium, Brussels, Oct. 2018), 386–393.

[6] Kingma, D.P. and Ba, J. 2014. Adam: A Method for Stochastic Optimization. arXiv.

[7] Mager, M., Çetinoğlu, Ö. and Kann, K. 2020. Tackling the Low-resource Challenge for Canonical Segmentation. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Online, Nov. 2020), 5237–5250.

[8] Mawere, S. and Wu, S. 2022. MORPH\_SEGMENT2. GitHub: <https://github.com/darthskyy/MORPH_SEGMENT2>. Accessed: 2022-11-10

[9] Mesham, S., Hayward, L., Shapiro, J. and Buys, J. 2021. Low-Resource Language Modelling of South African Languages. *CoRR*. abs/2104.00772, (2021).

[10] Moeng, T., Reay, S., Daniels, A. and Buys, J. 2021. MORPH\_SEGMENT: 2021. *https://github.com/DarkPr0digy/MORPH\_SEGMENT*. Accessed: 2022-08-27.

[11] Moeng, T., Reay, S., Daniels, A. and Buys, J. 2022. Canonical and Surface Morphological Segmentation for Nguni Languages. *Proceedings of The Second Southern African Conference for AI Research, 6-10 December 2021, Dolphin Coast, KwaZulu-Natal, Communications in Computer and Information Science, Springer* (Cham, 2022), 125–139.

[12] Narasimhan, K., Karakos, D., Schwartz, R., Tsakalidis, S. and Barzilay, R. 2014. Morphological Segmentation for Keyword Spotting. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Doha, Qatar, Oct. 2014), 880–885.

[13] Opidi, A., Jha, A. 2022. PyTorch Loss Functions: The Ultimate Guide: 2022. *https://neptune.ai/blog/pytorch-loss-functions*. Accessed: 2022-10-03.

[14] Pirrelli, V. 2019. Morphological Theory and Computational Linguistics. *The Oxford Handbook of Morphological Theory*. J. Audring and F. Masini, eds. Oxford University Press. 572–593.

[15] Roest, C., Edman, L., Minnema, G., Kelly, K., Spenader, J. and Toral, A. 2020. Machine Translation for English–Inuktitut with Segmentation, Data Acquisition and Pre-Training. *Proceedings of the Fifth Conference on Machine Translation* (Online, Nov. 2020), 274–281.

[16] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* 15, 1 (Jan. 2014), 1929–1958.

[17] Wu, S. and Cotterell, R. 2019. Exact Hard Monotonic Attention for Character-Level Transduction. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (Florence, Italy, Jul. 2019), 1530–1537.

[18] Wu, S., Cotterell, R. and Hulden, M. 2021. Applying the Transformer to Character-level Transduction. *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume* (Online, Apr. 2021), 1901–1907.

[19] Wu, S. 2022. neural-transducer: 2022. *https://github.com/shijie-wu/neural-transducer*. Accessed: 2022-08-27.

[20] Wu, S., Shapiro, P. and Cotterell, R. 2018. Hard Non-Monotonic Attention for Character-Level Transduction. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (Brussels, Belgium, Oct. 2018), 4425–4438.

[21] Xu, K., Ba, J.L., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R.S. and Bengio, Y. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37* (2015), 2048–2057.

[22] Yao Hengshuai and Zhu, D. and J.B. and Y.P. 2020. Negative Log Likelihood Ratio Loss for Deep Neural Network Classification. *Proceedings of the Future Technologies Conference (FTC) 2019* (Cham, 2020), 276–282.

[23] Zeiler, M.D. 2012. ADADELTA: An Adaptive Learning Rate Method. arXiv.