information

MDPI

Article

Viability of Neural Networks for Core Technologies for Resource-Scarce Languages

Melinda Loubser and Martin J. Puttkammer *

Centre for Text Technology, North-West University, Potchefstroom 2520, South Africa; Melinda.Loubser@gmail.com

* Correspondence: Martin.Puttkammer@nwu.ac.za; Tel.: +27-82-495-0609

Received: 15 November 2019; Accepted: 25 December 2019; Published: 12 January 2020

Abstract: In this paper, the viability of neural network implementations of core technologies (the focus of this paper is on text technologies) for 10 resource-scarce South African languages is evaluated. Neural networks are increasingly being used in place of other machine learning methods for many natural language processing tasks with good results. However, in the South African context, where most languages are resource-scarce, very little research has been done on neural network implementations of core language technologies. In this paper, we address this gap by evaluating neural network implementations of four core technologies for ten South African languages. The technologies we address are part of speech tagging, named entity recognition, compound analysis and lemmatization. Neural architectures that performed well on similar tasks in other settings were implemented for each task and the performance was assessed in comparison with currently used machine learning implementations of each technology. The neural network models evaluated perform better than the baselines for compound analysis, are viable and comparable to the baseline on most languages for POS tagging and NER, and are viable, but not on par with the baseline, for Afrikaans lemmatization.

Keywords: resource-scarce languages; core technologies; South African languages; neural networks; machine learning

1. Introduction

South Africa is a linguistically diverse country with at least 35 spoken languages, 11 of which are granted official status in terms of the Constitution of South Africa. Based on their orthographies, these official languages are all either Southern Bantu languages or West-Germanic languages, and can be categorized on a conjunctive-disjunctive scale into three groups:

(1) Conjunctive languages (four Nguni languages, viz. isiZulu, isiXhosa, isiNdebele, and Siswati);
(2) Disjunctive languages (Tshivenda, Xitsonga, and three Sotho-Tswana languages (Sesotho, Sepedi and Setswana)); and
(3) Middle of the scale (two West-Germanic languages, viz. Afrikaans and English.

The following example provided by [1] illustrates the difference between conjunctive and disjunctive languages: the phrase "I love him/her" is written as a single word, *ngiyamthanda*, in isiZulu, while it is written as four separate words in Sepedi, *ke a mo rata*.

Several legislative frameworks such as the National Language Policy Framework (https://www.gov.za/documents/national-language-policy-framework-final-draft), Language in Education Policy (https://www.gov.za/documents/language-education-policy-0), Language Policy Framework for South African Higher Education (https://www.gov.za/documents/language-policy-framework-south-african-

higher-education), and the use of Official Languages Act (12/2012) promote the use and development of the official languages, especially the previously marginalized Southern Bantu languages. The development of human language technologies (HLTs) for these languages could contribute significantly to multilingualism and language development and ensure that South Africans are not excluded from benefits of improved human-machine interaction. Over the past two decades the South African government, specifically the Department of Arts and Culture, and the Department of Science and Innovation, have continuously supported the development of HLTs through various projects that entailed the development of natural language processing (NLP) resources in the form of data, core technologies, and software (most of these resources are available from the South African Centre for Digital Language Resources (www.sadilar.org)).

Even though considerable effort from government, universities, and several private institutions and individuals have been made to develop HLT resources for South African languages, all of them (with the exception of English) are still considered resource-scarce with relatively little data that can be used to develop NLP applications and technologies. Most available datasets are limited to circa 40,000 to 60,000 tokens for a specific task. In addition, several core technologies using rule-based approaches as well as supervised machine learning [2] have been developed. The ongoing development and improvement of core technologies is an important step towards reaching the goals set by the abovementioned policies, while also being a prerequisite for downstream NLP systems, such as machine translation. It is, therefore, important that these development efforts are in line with international best practices and trends.

Internationally, the trend in language technologies has shifted from rule-based systems, which prevailed until the 1990s [3,4], to data-driven, statistical, or supervised machine learning (for ease of reference in this paper, "machine learning" will be used to refer to sequential machine learning methods, thus excluding neural networks) based methods such as Hidden Markov Models (HMMs), decision trees, and memory-based learning [5,6], and over the last decade has shifted again from machine learning systems to parallel processing using neural networks [6–8]. As early as 2011, Collobert et al. [7] demonstrated that neural networks could outperform other machine learning methods available at that time on POS tagging, phrase chunking, NER, and semantic role labelling. Since 2013, Neural Machine Translation (NMT) [9–11] has established itself as the new state-of-the-art machine translation. NMT systems have achieved improvements of up to 20 BLEU points [12] over statistical machine translation systems [13], and is in use by technology giants like Google [14] and Facebook [15]. Neural models similar in architecture to NMT models have been used for other NLP tasks, such as lemmatization [16] and morphological reinflection [17]. For all NLP tasks, the main advantage of neural methods is that they learn feature representations internally, thereby precluding the necessity for task specific feature engineering [18].

Work on core technologies for South African (SA) languages has followed the same trend from rule-based to machine learning based systems. For example, a rule-based lemmatiser for Afrikaans, developed in 2003, was supplanted in 2007 by a memory-based machine learning model [19]. For isiXhosa, a machine learning based lemmatiser was developed in 2015 [20] using data annotated by a rule-based lemmatiser, developed in 2014 [2]. A rule-based compound analyser was developed in 2004 [21], followed by a machine learning based method in 2008 [22]. Several other machine learning algorithms are currently in use in core technologies for SA languages. HMMs were implemented for POS tagging for all ten SA languages [2,23], and a transition-based tagger was also evaluated [24]. Conditional Random Fields (CRFs) have been employed with reasonable success (average F1-score of 0.73) for NER for all ten SA languages [25]. Decision trees were used for automatic compound analysis of Afrikaans in [21]. Later work on compound analysis used *TiMBL*, a memory-based learner which employs the k-Nearest Neighbor (k-NN) algorithm [22]. The k-NN algorithm was also used to develop the *Lemma Identifier for Afrikaans* (Lia) [19,26]. A similar k-NN lemmatiser was tested for Setswana [27].

While much work has been done in moving from rule-based to machine learning-based systems for South African languages' technologies, almost no work (to the best of our knowledge, the only

exceptions are Fick, 2003 [28] and 2013 [29]) has been done on using neural networks instead of machine learning methods for core language technologies. A disadvantage of neural networks is that they generally require large amounts of data to work well [30], meaning that research on neural methods for mainstream languages like English cannot be assumed to apply directly to low-resource contexts. Research on neural networks for NLP in low-resource settings has recently gained some momentum (e.g., [31–33]), but South African languages have not been among those included in these studies. However, the promising performance of neural methods for core technologies like POS tagging and NER in high resource settings [7] and in low-resource settings for other languages [32,33] suggests that it is worth assessing their performance in the South African context.

To establish the viability of neural network methods for core language technologies for ten low-resource South African languages, we consider two sequence tagging tasks, POS tagging, and NER, and two sequence translation tasks, compound analysis and lemmatization (described in the next section). We select neural architectures that show most promise for each task and perform hyper-parameter optimization for each task. Neural network models are trained for each task and each applicable language (for compound analysis, this is limited to Afrikaans) and the neural models' performance is compared with that of the current machine learning based technologies.

2. Neural Network Architecture

All the tasks considered in this paper involve modelling sequential data. POS tagging and NER are sequence tagging tasks, while compound analysis and lemmatization can both be modelled as sequence-to-sequence (or sequence translation) tasks. Sequence tagging is a classification task, where the set of possible tags is discrete, and each token requires exactly one tag. Sequence translation requires the prediction of an output sequence based on the input sequence, where the input and output sequences could use different vocabularies and be of different lengths. The neural architectures considered in this paper are, therefore, limited to those aimed either at sequence tagging or sequence-to-sequence tasks.

2.1. Embeddings

For both sequence-to-sequence and sequence tagging tasks, feature representation for real-world machine learning models is challenging. One-hot encoding (for example, to represent the subsequent letter in a word, a one-hot encoded feature vector would represent all letters in the alphabet, and in every column except the one representing the subsequent letter, the value for those features would be zero. Hence, one-hot encoded vectors are sparse vectors) linguistic features like prefixes, suffixes, or sentence context as discrete features in a fixed-length vector results in sparse, high-dimensional input vectors [34]. The advantage of neural sequence tagging and sequence translation models is that they can learn feature representation internally, using word or character embeddings [35,36]. Word embeddings are continuous dense vector representations of words learned from the context in which a word appears in a large, monolingual corpus. Word embeddings can encode both semantic and syntactic similarities, such that simple algebra with the learned vectors yields meaningful results [37]. Character embeddings [38] can be learned with a similar approach, using characters as the individual tokens to be represented and thus capturing morphological information [8]. They can be used by themselves or in conjunction with word embeddings [39–41]. Character and word embeddings can be pre-trained using independent networks or learned as part of a task-specific model. The encoder portion of a neural machine translation model is one example of a model whose first layers learn word embeddings. For our experiments, we use pre-trained word embeddings to train models for NER and POS tagging.

2.2. Sequence Tagging

Sequence tagging neural networks for POS tagging and NER require only an encoder and a classifier. Many studies focus on the encoder portion of the model. Approaches include using convolutional neural networks (CNNs) [42], biLSTM's [43,44], and combinations of the two [40,45,46].

Using a combination of character and word-level word embeddings was found to improve performance on POS tagging [40] and NER [47]. Dos Santos and Guimarães [47] additionally found that the same network architecture and hyperparameters can be used to achieve state-of-the-art performance on both POS tagging and NER. Several studies looking at neural sequence tagging in a low-resource setting use some form of cross-lingual transfer learning, where the tags from a high-resource language are used to infer tags for a low-resource language using a parallel corpus [48] or bilingual dictionary [49]. The addition of only a small amount of tagged monolingual data in the target language results in a large boost in performance to transfer-learning models [48,50] suggesting that using monolingual data remains preferable, if at all possible. Plank et al. [51] tested their implementation of a character-sensitive biLSTM tagger, *biLSTM-aux*, in a low-resource setting without transfer learning and found that their model performed better than expected even with only 500 tagged sentences of training data. Their model outperformed HMM-based and CRF-based POS taggers on twenty-one languages (none of which are South African languages), including Slavic and Nordic languages. However, they only included languages with more than 60,000 tokens, which is more than is available for most South African languages. Our work in this paper follows on theirs as we evaluate the viability of their model for POS tagging and NER for South African languages.

2.3. Sequence-to-Sequence

Sequence-to-sequence neural networks were first proposed for neural machine translation (NMT) [11] using an encoder-decoder architecture as described in [9,10]. The now well-established basic model for NMT consists of a bidirectional Recurrent Neural Network (bi-RNN) to encode the sequence into a fixed-length vector representation and a decoder RNN that predicts the probability of a target sequence based on that representation. The decoder uses both the state in the decoder RNN and a context vector, which serves as an alignment or attention mechanism [11] to predict the output sequence. A significant improvement to this model was the substitution of the RNNs with Long-Short Term Memory (LSTM) cells, which are better suited to dealing with long range dependencies [10]. The massive success of encoder-decoder models for NMT led to research on their application to character sequence translation tasks. Faruqui et al. [52] and Kann and Schütze [17] used variations on encoder-decoder architectures for morphological inflection generation. In both studies, the neural models achieved results comparable or superior to non-neural state-of-the art methods. Schnober et al. [53] evaluated the encoder-decoder approach for four-character sequence translation tasks and found that non-neural algorithms outperformed sequence-to-sequence encoder-decoder models on three of the four tasks, except for lemmatization. They hypothesized that neural sequence-to-sequence models are well suited to lemmatization precisely because of the presence of long-range dependencies, which are mostly absent in the other three tasks (OCR post correction, spelling correction, and grapheme-to-phoneme conversion). A context-sensitive neural lemmatiser, *Lematus* [16], achieved an average accuracy of 94.9% across 20 languages in a high-resource setting, and 87.8% accuracy in a simulated low-resource setting. A point of interest in this study is their finding that lexical ambiguity, morphological productivity, and morphological regularity are better predictors of performance than the amount of training data available for a given language. This is relevant for the experiments in this paper since several South African languages are agglutinative and highly morphologically productive. For our lemmatization experiments, we use an implementation of *Lematus*.

There exists only limited published research on neural models for compound analysis. Fick [29] used a windowing approach and a multilayer-perceptron (MLP) network for compound splitting in Afrikaans, modelling the task as a classification problem. Their MLP network achieved a maximum accuracy of 86% at the word level. Hellwig [54] used RNNs to model compound analysis and Sandhi resolution in Sanskrit as a sequence tagging task, where the target tags are transformation rules. Dima and Hinrichs [55] used deep neural networks for compound interpretation in English, which is a semantic task, unlike compound analysis. To the best of our knowledge, there has not been any work done on compound analysis as a sequence-to-sequence problem, but work on the similar tasks of

morphological analysis and lemmatization suggest that encoder-decoder networks are a valid approach to the problem. For our compound analysis experiments, we use a straightforward character-based biLSTM sequence-to-sequence model.

3. Method

3.1. Languages and Data

The four technologies addressed in this paper were selected partly because they represent two categories of tasks—sequence tagging and sequence translation—and partly because annotated data is available for them. The availability of data and machine learning systems for comparison likewise determined which languages were included for each task. Even though the scope of this study is restricted to South African languages, the results should not only be applicable to them, but also to other languages from similar language families.

3.1.1. Sequence Tagging

Training data statistics for the sequence tagging tasks are given in Table 1. POS-tagging models were trained for all ten languages using the data from the NCHLT Text project [2] (datasets and tagging protocols are available at https://repo.sadilar.org/handle/20.500.12185/1). The datasets were split into training and validation sets (90% and 10%, respectively) for hyper-parameter selection, and the separate 5000-token test sets were held out for final model evaluation. For Afrikaans, one model was trained using only the NCHLT data and another was trained using an additional 50,000 tokens from [56]. The number of tokens available for a language partially depends on the writing system of that language. All datasets are parallel, based on the same English dataset, but conjunctively written (henceforth conjunctive) languages by nature use fewer orthographic words (tokens) to express the same idea as a disjunctively written (henceforth disjunctive) language. Therefore, the disjunctive languages have around 60,000 tokens available each, while conjunctive languages only have around 40,000. See [57] for a more detailed discussion of disjunctively vs. conjunctively written Bantu languages. The number of unique tags refers to the number of unique tags found in the training data for each language.

Table 1. POS tagging and NER training data statistics.

Language	Writing System	Tokens (POS)	Types (POS)	Unique Tags in Data (POS)	Tokens (NER)	Types (NER)	Named Entities (NER)
Afrikaans (af)	Mixed	55,483	7108	98	206,614	22,657	12,543
Af-100k	Mixed	100,423	12,470	113	–		–
IsiNdebele (nr)	Conjunctive	38,426	13,558	95	145,190	40,046	15,854
IsiXhosa (xh)	Conjunctive	42,061	15,008	75	108,766	35,622	12,260
IsiZulu (zu)	Conjunctive	41,714	14,125	99	180,751	52,628	18,592
Siswati (ss)	Conjunctive	39,486	13,628	92	157,412	45,013	16,429
Sepedi (nso)	Disjunctive	62,841	5782	137	181,299	16,822	10,583
Sesotho (st)	Disjunctive	62,929	5831	165	242,148	16,817	12,530
Setswana (tn)	Disjunctive	62,842	6024	139	209,085	17,234	11,498
Xitsonga (ts)	Disjunctive	62,961	5828	143	240,937	16,600	14,770
Tshivenda (ve)	Disjunctive	59,818	5340	220	211,694	15,059	10,701

The data for the NER task consists of data from the NCHLT Text Project and additional government-domain data as described in [25]. The data is tagged according to the CoNLL-2003 shared task protocol, which specifies that tokens are either at the beginning of (B), inside (I), or outside (O) a named entity and also specifies whether the entity is a person (PERS), organization (ORG) or location (LOC) [58]. The NER datasets were each split into training (80%), validation (10%), and test (10%) sets. NER models were trained for all ten languages.

3.1.2. Sequence Translation

Compound analysis experiments were carried out for Afrikaans only, since neither annotated data nor automatic compound analysers are available for languages besides Afrikaans. The CKarma

dataset [21], consisting of 77,850 Afrikaans words annotated with compound boundaries, was used for these experiments. The data was split into training, validation, and test splits (80%, 10%, and 10%, respectively) and the test split was held out for final model evaluation.

Lemmatisation experiments were carried out for all ten languages. The lemmatization models were trained using the training data from the NCHLT Text Project. For Afrikaans, the data annotated during the development of Lia [19] (see Section 2) was also used. The neural lemmatization model is context sensitive if context is provided, as it is in the NCHLT corpora. On the other hand, Lia is a context-insensitive lemmatiser and the data annotated for that project consists of a list of unique words without context. Therefore, to permit both direct comparison with Lia and evaluate the effect of using sentence context, a context-sensitive model was trained using the NCHLT data and a context-insensitive model was trained using the training data from Lia. The NCHLT data was split into training and validation sets (90% and 10%, respectively) and the validation set was used for hyper-parameter selection. The test set developed during the NCHLT Text Project was held out for final evaluation of both models. Training data statistics for lemmatization are given in Table 2.

Table 2. Train and test data statistics for lemmatization.

Language	Tokens	Types	Lemmas	Tokens	Types	Lemmas
af-lia	72,226		61,881	–		–
af-nchlt	55,483	7108	5515	5834	1675	1450
nr	38,426	13,558	3595	3904	2323	1636
nso	62,841	5782	2866	7153	1485	1042
ss	39,486	13,628	7162	4075	2364	1813
st	62,929	5831	2810	6847	1563	1080
tn	62,842	6024	2970	6803	1588	1144
ts	62,961	5828	4542	6518	1450	1243
ve	59,818	5340	3606	6646	1493	1245
xh	42,061	15,008	3577	4409	2547	1707
zu	41,714	14,125	2830	4343	2415	1647

3.2. Models

For each task, we use the same hyperparameters which were tuned using the Afrikaans validation data. Although it would be ideal to tune hyperparameters per task and language, time, and resource constraints did not allow for this. However, experiments using Setswana and isiZulu data and varying the hyperparameters for POS tagging indicated that very similar hyperparameters performed well across languages. When training a model for each language, the number of epochs (iterations) trained through was adjusted using early stopping based on performance on the validation set. When the score for the model's predictions on the validation set do not improve for a certain number of epochs, training stops and the model from the last epoch is saved. The final model for each language and task was trained using both the training and validation data and was evaluated on the held-out test set.

3.2.1. Sequence Tagging

We followed [47] in using the same model architecture and parameters for both POS tagging and NER. The sequence tagging models used Plank et al.'s [51] implementation of a hierarchical bidirectional LSTM with an auxiliary loss function which they called *bilstm-aux* (https://github.com/bplank/bilstm-aux). The first level learns word embeddings and sub-token embeddings, and these vectors are concatenated to form the input to the higher-level LSTMs. The model is trained to predict both tags and the log frequency of the source token, a strategy intended to discourage learning shared representations between rare and common words, thus improving the handling of rare words. *bilstm-aux* deals with token and sub-token representation internally given a file in CoNLL-U format [59]. Word vector size, hidden layer size and number, and learning rate were tuned by varying one parameter at a time, starting with number of hidden layers and using Plank et al.'s reported settings as start

values. The final model had four layers and word vectors in dimension 200 (except when pre-trained embeddings were used), character vectors in dimension 100, and 400 hidden units. The model was trained using a learning rate of 0.1, the SGD optimizer, and regularization by word dropout of 0.25. We also trained models using the *fasttext* (https://fasttext.cc/docs/en/pretrained-vectors.html) [39], embeddings trained on a Wikipedia Corpus (XML Wikipedia dumps: 11 September 2017) [60], in dimension 300, to initialize the word embeddings for all languages except isiNdebele, for which *fasttext* embeddings are not available.

3.2.2. Sequence Translation

The framework for our compound analysis model is an LSTM-RNN encoder-decoder model [9,10] with two layers in the encoder and decoder, using global attention and beam-search in translation [61]. The input is a sequence of characters and the target output is a sequence of characters interpolated with compound boundaries (+) and valence morpheme boundaries (_) as seen in Table 3. This format is similar to that described in [22], with the exception that the sequence is space separated and is appended with an end-of-sequence (.) marker, which preliminary experiments indicated improved the accuracy of sequence length prediction. As with morphological inflection generation [52], the compound analysis task differs from neural machine translation in that the output sequence is very similar to the input sequence, except for the boundary markers. Therefore, we also incorporated a copying mechanism [62] by which infrequently seen tokens in the input are copied directly to the output sequence. The frequency limit below which copying is required was tuned as one of the hyperparameters of the model. Although [10] found that reversing the input helped, preliminary experiments with the compound analysis data did not indicate any increase in performance by doing so. The model was implemented in OpenNMT, an open-source neural machine translation toolkit [63]. Hyperparameters were selected by varying one at a time while holding others constant, using the guidelines suggested in [64]. The hyperparameters varied during hyper-parameter optimization included: the learning rate, decay rate, word vector size, number of hidden units, optimization algorithm, the minimum token frequency below which the source token would be copied to the predicted sequence, and the settings for scheduled sampling [65]. Scheduled sampling allows the decoder to use the gold reference token instead of the previously generated token according to a probability that decreases (decays) over training epochs, so that errors in the beginning of a prediction will not be propagated to the rest of the sequence during early epochs. It was found that a learning rate of 0.001 using the Adam optimization algorithm and a decay rate of 0.9 with a word vector size of 500 and 500 hidden units yielded the best results. It was further found that applying scheduled sampling with an initial probability of 0.5 and a linear decay rate of 0.01 increased the performance of the trained model.

Table 3. Input and Target format for the compound analysis model for the word "regeringsbeleid" ("government policy").

Input Sequence	Target Sequence
r e g e r i n g s b e l e i d	r e g e r i n g _ s + b e l e i d.

Our lemmatization model is based on the *Lematus* lemmatiser described in [16]. The network is a two-layered attentional bidirectional encoder-decoder network using GRU cells in all layers; the only adaption we made to the *Lematus* architecture is that a non-conditional GRU was used in both layers of the decoder. The input to the model is a space separated character sequence of the target word and its left and right sentence context of size n. No padding characters are used when there are less than n characters in the sentence context to the left or right of the word. The left and right context, word, and beginning and end of phrase boundaries are marked with the tokens (<lc> <rc>), (<s>) and (<w> <\w>) respectively (see Table 4 for an example of how these markers are used). The model learns representations for these markers exactly as it does for other characters. An example of the input and

target sequence format is shown in Table 4 for a left and right context size of ten characters. If n = 0, the model is context-insensitive, and the input includes only the source token. Hyperparameters were selected by varying one at a time while holding others constant, starting with the parameter's settings described in [16]. In the final model, a context size of 25 with length normalization of 0.7 was used, 300 units in the hidden layer, word vectors in dimension 500 and scheduled sampling at an initial probability of 0.5. The model was trained using the Adadelta optimization algorithm and word dropout of 0.4 for regularization. Words appearing with a frequency less than 100 in the training corpus were dealt with as unknown tokens. The lemmatization model was also implemented in OpenNMT.

Table 4. Input and target format for the lemmatization model with context size = 10 for the Afrikaans sentence "Laai die elektroniese aansoekvorm af" ("Download the electronic application")].

Input Sequence	Target Sequence
<w> <lc> L a a i <rc> d i e <s> e l e k t r <\w>	l a a i
<w> L a a i <lc> d i e <rc> e l e k t r o n i e <\w>	d i e
<w> L a a i <s> d i e <lc> e l e k t r o n i e s e <rc> a a n s o e k v o r <\w>	e l e k t r o n i e s
<w> e k t r o n i e s e <lc> a a n s o e k v o r m <rc> a f <s> <\w>	a a n s o e k v o r m
<w> a a n s o e k v o r m <lc> a f <rc> <s> <\w>	af

3.3. Evaluation

3.3.1. Sequence Tagging

The performance of the neural models is evaluated against baselines, as achieved by current machine learning implementations of each technology. For POS tagging, the baseline system is the Mate Tools POS tagger [66,67] which was found by [24] to achieve the highest average accuracy across all ten languages. For NER, we compare results with those reported in [25] as a baseline, using precision, recall, and F-score at named entity levels as metrics. NER entails both demarcation of named entity boundaries and assigning a type to predicted named entities. Tokens are designated as either at the beginning of (B), inside (I), or outside (O) of a named entity, and "B" or "I" tagged tokens are further tagged as either a person (PERS), organization (ORG), or location (LOC). Several methods for determining what counts as "correct" (true positive) classifications of NERs exist. For our evaluation, the metrics are calculated as per the method defined for the CoNLL-2003 Shared Task [58], which uses the strictest definition of a correct prediction to determine true positives, excluding any partial predictions of named entities and errors in the type of named entity predicted.

3.3.2. Sequence Translation

For compound analysis, results are compared with those reported in [22] as a baseline, which were achieved using a memory-based learner. They are additionally compared with those achieved by *CatBoost*, a gradient-boosted decision tree (GBDT) implementation [68], which was found to be the best-performing machine learning algorithm of those evaluated by the authors (machine learning methods tested included kNN's (IBk), non-boosted decision trees, Random Forest, SVM's, One Rule, and Naïve Bayes). The comparison metrics for compound analysis are accuracy at word-level and F1-score at the compound boundary level. For the lemmatization task, the baseline for Afrikaans is Lia (see Section 2) [19] and precision, recall, F1 and accuracy are reported. Tokens consisting of punctuation or numbers were excluded from evaluation, and for systems trained using the Lia data, all tokens were lowercased before prediction, since Lia does not predict capitalization. For languages other than Afrikaans, the only available lemmatizers are rule-based. Therefore, the lemmatization experiments on other languages lack a machine learning baseline for comparison and results are reported with the rule-based scores alongside for interest only.

4. Results and Discussion

4.1. POS Tagging

Our results for POS tagging are given in Table 5. Overall, *Mate* achieves the highest average accuracy, followed closely by *biLSTM-aux* with embeddings and *biLSTM-aux* alone. Although using word embeddings slightly improved the average accuracy of the neural model, it outperformed *biLSTM-aux* alone on only four languages, with the greatest improvement for Afrikaans. This result is expected given the size and quality of the Wikipedia corpora used to train the fasttext embeddings. Both neural models fall short of Mate on all languages except Afrikaans, Sepedi, and Sesotho. Notice that the languages on which the neural models outperform *Mate* are all disjunctive languages, except for Afrikaans (Afrikaans orthography falls between conjunctive and disjunctive), and the performance of all models on conjunctive languages is lower than on disjunctive languages (cf. Table 1 for orthography type). At first glance, this might appear to be related to the amount of training data or the number of tags for each language.

Table 5. POS tagging accuracy; numerically highest scores per language are in bold font.

	Mate Baseline	bilstm-aux	bilstm-aux with Fasttext Embeddings	BiLSTM-aux with 40,000 Tokens
English *	92.66% (TnT)	92.10%	**95.16%**	n/a
German *	92.64% (TnT)	90.33%	**93.38%**	n/a
af-100k	91.50%	91.50%	**92.00%**	n/a
af	93.90%	93.10%	**94.30%**	91.20%
nr	**83.00%**	79.90%	n/a **	79.90%
nso	94.80%	**95.20%**	94.90%	94.10%
ss	**83.00%**	81.40%	81.50%	81.40%
st	89.40%	89.10%	**89.90%**	88.20%
tn	**90.10%**	89.20%	88.90%	88.50%
ts	**88.60%**	88.30%	88.10%	87.30%
ve	**87.40%**	86.50%	86.00%	85.40%
xh	**87.60%**	84.90%	85.30%	84.90%
zu	**85.90%**	84.10%	84.00%	84.10%
Average (SA languages)	**88.37%**	87.17%	88.10%	86.50%

* As reported in Plank et al. [51] for the Universal Dependencies v1.2 datasets. ** No pre-trained embeddings are available for isiNdebele.

To assess the effect of training corpus size, *biLSTM-aux* models for the disjunctive languages were also trained on reduced datasets of approximately 40,000 tokens, which is the amount available for conjunctive languages. In this setting, accuracy was still an average of 6% higher for disjunctive languages verses conjunctive languages. A likely explanation is that conjunctive languages' words are generally much longer and more morphologically complex, resulting in more word types, and therefore less examples per type, for a corpus of the same size in comparison with disjunctively written languages [57]. For disjunctive languages, despite having larger tag set sizes than conjunctive languages, the ratio of tokens to types is more than 10:1; for conjunctive languages, the ratio is less than 3:1. Orthography type is thus a significant factor for a data-driven POS tagging system, including neural networks.

The accuracies reported in [51] for English and German POS models are recorded in Table 5 as examples of results achieved for high-resource languages. The models were trained on more than 250,000 tokens and used the Polyglot pre-trained embeddings [69]. For Afrikaans, a larger dataset of approximately 100,000 tokens was also available. The performance of *biLSTM-aux* trained on the 100,000 Afrikaans dataset is lower than for just the NCHLT dataset, a result which can be explained by the domain of the extra data. While the NCHLT data comes from the government-domain only, the

additional 50,000 tokens come from multiple domains, meaning more variation in training and testing examples. The German results are most comparable with the Afrikaans 100,000 results, in terms of both language type and domain of data. It is interesting that *biLSTM-aux* alone performs better on Afrikaans 100,000 than on German, while the addition of pre-trained embeddings is more helpful for German than Afrikaans (3% and 0.5% improvement, respectively). This result suggests that the size and quality of the pretrained embeddings is a significant factor in the performance of the neural POS tagger.

4.2. NER

Results for NER experiments are given in Table 6. The CRF baseline performs best on four languages and the neural models outperform it on five languages, while the scores for Afrikaans are almost identical (within 0.01%). There is no apparent trend according to language type. The neural models' largest problem is low precision, in contrast to the CRF model, whose weakness is low recall, as noted in [25]. CRF outperforms both neural models on precision on eight languages and is outperformed by biLSTM-aux on recall on seven languages. Both neural models' performance on Sesotho is significantly lower than on the other languages, falling short of the CRF model by more than 18%. A closer analysis of the prediction output showed that most incorrect predictions were false positives where the prediction of a named entity was entirely false, not just incorrect in type or boundary. The predominance of this kind of error is in line with the trend across all languages, but it is not clear why the rate is so much higher for Sesotho, especially given that the CRF classifier performed fairly well on it, suggesting that the data itself is not inherently the problem. If Sesotho is excluded as an outlier, the neural models' average F-score is 73.53%, which is slightly higher than the CRF baseline at 73.24%.

Table 6. NER F1-scores; numerically highest precision, recall and F1 scores per language are in bold font.

	CRF Baseline *	bilstm-aux	bilstm-aux emb	CRF Baseline *	bilstm-aux	bilstm-aux emb	CRF Baseline *	bilstm-aux	bilstm-aux emb
	Precision			Recall			F1		
af	**78.59%**	73.61%	73.41%	73.32%	**78.23%**	**78.23%**	**75.86%**	75.85%	75.74%
nr	77.03%	**78.58%**	n/a **	73.26%	**79.20%**	n/a **	75.10%	**78.89%**	n/a **
nso	**76.12%**	75.91%	72.14%	72.88%	**79.66%**	77.63%	74.46%	**77.74%**	74.79%
ss	69.03%	**70.02%**	69.93%	60.17%	71.44%	**72.82%**	64.29%	70.72%	**71.35%**
st	**76.17%**	53.29%	50.31%	**70.27%**	55.56%	57.73%	**73.09%**	54.40%	53.77%
tn	**80.86%**	74.14%	73.45%	75.47%	**77.42%**	74.71%	**78.06%**	75.74%	74.07%
ts	**72.48%**	72.33%	71.03%	69.46%	**71.44%**	71.25%	70.93%	**71.88%**	71.14%
ve	**73.96%**	67.97%	63.82%	**72.92%**	65.91%	67.09%	**73.43%**	66.92%	65.41%
xh	**78.60%**	69.83%	69.08%	**75.61%**	73.30%	72.78%	**77.08%**	71.52%	70.88%
zu	**73.56%**	72.43%	73.44%	66.64%	72.64%	**74.32%**	69.93%	72.54%	**73.87%**
Average	**75.64%**	70.81%	68.51%	71.00%	**72.48%**	71.84%	**73.22%**	71.62%	70.11%

* As reported in [25] ** No pre-trained embeddings are available for isiNdebele.

4.3. Compound Analysis

Table 7 shows that the neural compound analysis model improved over previously published results by a very large margin and outperformed the CatBoost model by a smaller margin on both accuracy and F1 score. Precision and recall are fairly balanced.

Table 7. Afrikaans compound analysis accuracy (word level) and F1-scores (constituent boundaries); numerically highest scores are in bold font.

	Knn Baseline	CatBoost *	Neural Model
Accuracy	81.28%	92.50%	**96.13%**
Precision	Not available	96.84%	**98.05%**
Recall	Not available	96.22%	**98.27%**
F1-score	90.57%	96.53%	**98.16%**

* Results from 10-fold cross-validation.

A qualitative analysis of the neural model's output shows that it makes some unique errors not made by classification algorithms, as seen in Table 8. Classification algorithms predict only one of a closed set of the categories for an instance, leading to errors in presence, absence, kind, and position of boundary markers. In the neural model, however, all items in the vocabulary are candidates for prediction at every point in the sequence, leading to errors in word spelling. In total, such errors were present in 21 predictions of the test set (less than 0.003% of instances). While infrequent, their occurrence indicates a disadvantage of sequence-to-sequence learning for tasks where the target is very similar to the input, or where only a closed class of changes to a sequence are possible.

Table 8. Examples of erroneous analyses made by the neural compound analysis model.

Error	Word	Predicted Analysis	Correct Analysis
Adding a character	Mosambiek	Mosamb**b**iek	Mosambiek
Deleting a character	teboekstelling	te+oek+stelling	te+boek+stelling
Substituting a character	lintbebouing	lint+b**b**bouing	lint+b**e**bouing

As per the findings in [53], neural networks have the greatest advantage over other machine learning methods for sequence translation tasks when long-range dependencies are involved. Therefore a probable explanation for the large improvement of the neural model over the baseline is that the use of character embeddings to represent characters, and the automatic combination of these embeddings into more complex features in the subsequent layers of the neural network, leads to a much more informative representation of the sequence at hand than does the windowing approach used by the kNN baseline. However, this benefit is not seen as clearly in the other tasks assessed. Two explanations present themselves. Firstly, a fairly large amount of data was available for training the model, and this data consisted only of informative examples, not running text with many non-compound words, in contrast with the data available for the other tasks. Secondly, compound analysis is a comparatively simple task. POS-tagging and NER are multiclassification tasks with many possible tags, and lemmatization is an open-ended sequence translation task, while compound analysis requires only the placement of two kinds of boundary markers. Establishing more firmly the reasons for the success of neural networks on this task would be a worthwhile direction for further research.

4.4. Lemmatisation

The results for the lemmatization experiments (see Table 9) are not directly comparable across models except for Afrikaans (see Section 3.3). For the models trained on the Afrikaans NCHLT data, the context-insensitive model outperforms the context-sensitive model, which is in line with the findings in [16].

Table 9. Lemmatisation results for Afrikaans.

	Lia (Lia Data) Baseline	Context-Insensitive Neural Model (Lia Data)	Context-Insensitive Neural Model (NCHLT Data)	Context-Sensitive Neural Model (NCHLT Data)
Accuracy	95.92%	94.84%	95.10%	93.68%
Precision	90.74%	78.39%	82.63%	77.75%
Recall	86.46%	88.99%	95.97%	95.83%
F1-score	88.55%	83.35%	88.80%	85.85%

The context-insensitive neural model trained on the Lia data falls roughly 5% short of Lia. This gap is accounted for entirely by false positives (low precision). Just over half of the false positives were cases where the model lemmatized the word by removing one or two letters from the front or back of the word, a kind of error made by Lia, too, albeit less frequently, since the majority of lemmatization rules are of that form. However, a few of the false positives are accounted for by errors

of the kind seen in Table 8 for the compound analysis. These are difficult to quantify since not all incorrect deletions, additions, or substitutions of characters in the middle of words are implausible or unseen lemmatization processes, as they are with compound analysis. The clearest cases were eight very long words (length > 20), such as "gemeenskapsveiligheidsforum" (community safety forum), which the neural model shortened by removing several characters from the middle of the word (e.g., "gemeenskapsveidsforum"). Unsurprisingly, the context-insensitive model trained on NCHLT data performs much better than the one trained on Lia data. This is most likely due to the ratio of training examples per lemma present in the training data, which is much higher in the NCHLT data than in the Lia data, and also due to the NCHLT data being in-domain relative to the test data, unlike the Lia data.

The scores for lemmatization on the other SA languages are reported in Table 10. As with POS tagging, the neural models generally perform better on disjunctive languages than conjunctive languages. With the exception of Xitsonga, the context-insensitive models outperform the context-sensitive models. There is a large discrepancy between precision and recall for the neural models. The average precision of the context insensitive model is 70.56%, while the average recall is 96.41%. Recall exceeded 99% on four languages. Thus, the major challenge for the neural model is low precision.

Table 10. Lemmatisation F1 scores on 9 SA languages; numerically highest scores per language are in bold font.

	Rule-Based * Baseline	Context-Insensitive Neural Model	Context-Sensitive Neural Model
nr	**80.32%**	73.17%	67.39%
nso	77.90%	**91.16%**	88.59%
ss	**81.60%**	79.61%	69.51%
st	76.43%	**82.58%**	80.86%
tn	74.86%	**76.70%**	73.76%
ts	76.09%	85.67%	**86.50%**
ve	77.54%	**79.68%**	79.34%
xh	**79.82%**	74.92%	74.10%
zu	**81.56%**	74.87%	74.08%
Average	78.46%	**79.82%**	77.13%

* As reported in [2].

5. Conclusions and Future Work

In this paper we trained and evaluated neural network implementations of core language technologies for ten low-resource South African languages. For POS tagging, results show that the neural model performs comparably with the baseline on Afrikaans and disjunctive languages (accuracy within 1%), and slightly worse on conjunctive languages, falling short of the baseline by 2.3% on average. In addition, the quality of the pretrained embeddings available significantly affects the performance of the neural POS tagger, as their use improves performance only for Afrikaans. These results indicate that the neural model evaluated is viable for POS tagging for SA languages, but is not superior to the machine learning baseline. The neural NER model performs on par with or better than the baseline on most languages, with no trend according to orthography type. There is, however, considerable variation in performance per language, with one extreme outlier (Sesotho) which falls short of the baseline by 18% (F-score). Thus, neural networks are a viable implementation for NER for most SA languages, but further research on the variation in performance per language is needed to ensure viability for all languages. For lemmatization, the best neural model is a context-insensitive model. In the only directly comparable case (Afrikaans), the machine learning baseline outperforms the neural model by 5%. However, the training data used significantly impacts the performance of the neural model, indicating that the implementation itself is viable, but depends on the training data

type. On the other nine languages, there is a strong trend in performance according to orthography type, with higher performance on disjunctive languages than on conjunctive ones. Due to the lack of machine learning baselines for the other languages, it is not possible to directly assess the viability of the neural models in comparison to machine learning systems. The neural compound analysis model outperforms both machine learning baselines on all metrics, achieving a word-level accuracy of 98%. Of the technologies assessed, the compound analysis task is the only one on which the neural model evaluated is clearly superior to the baseline, probably because of the presence of long-range dependencies, the amount and type of data available for the task, and the simplicity of the task in comparison with lemmatization, POS tagging and NER.

Overall, our results demonstrate that neural networks can be viable implementations of core language technologies for resource-scarce South African languages. However, the implementations assessed in this paper are not generally superior to current machine learning baselines, except for compound analysis.

The experiments in this paper evaluated only one neural architecture for each task. Therefore, future work should consider alternative neural architectures, such as transformers, and parameters. Given the unique errors introduced by modelling compound analysis and lemmatization as sequence translation tasks, it would be worth investigating neural models for lemmatization and compound analysis modelled as classification tasks. Another open question for future research is the viability of the neural models in terms of computational resources and efficiency in the context of their use in downstream tasks. Future research could also extend the assessment of neural network implementations to other core technologies such as morphological analysis and phrase chunking.

Author Contributions: Conceptualization: M.J.P. and M.L.; data curation: M.J.P. and M.L.; formal analysis: M.L.; funding acquisition: M.J.P.; investigation: M.L.; methodology: M.J.P. and M.L.; project administration: M.J.P.; resources: M.J.P.; software: M.J.P. and M.L.; supervision: M.J.P.; validation: M.J.P. and M.L.; visualization: M.J.P. and M.L.; writing—original draft: M.L.; writing—review and editing: M.J.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was made possible with the support from the South African Centre for Digital Language Resources (SADiLaR). SADiLaR is a research infrastructure established by the Department of Science and Technology of the South African government as part of the South African Research Infrastructure Roadmap (SARIR).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Prinsloo, D.J.; De Schryver, G.M. Towards an 11 × 11 array for the degree of conjunctivism /disjunctivism of the South African languages. *Nord. J. Afr. Stud.* **2002**, *11*, 249–265.
2. Eiselen, R.; Puttkammer, M.J. Developing Text Resources for Ten South African Languages. In Proceedings of the Ninth International Conference on Language Resources and Evaluation, Reykjavik, Iceland, 26–31 May 2014; pp. 3698–3703.
3. Marcus, M. New trends in natural language processing: Statistical natural language processing. *Proc. Natl. Acad. Sci. USA* **1995**, *92*, 10052–10059. [CrossRef] [PubMed]
4. Liberman, M.Y. The Trend towards Statistical Models in Natural Language Processing. In *Natural Language and Speech*; Klein, E., Veltman, F., Eds.; Springer: Berlin/Heidelberg, Germany, 1991; pp. 1–7.
5. Nadkarni, P.M.; Ohno-Machado, L.; Chapman, W.W. Natural language processing: An introduction. *J. Am. Med. Inform. Assoc.* **2011**, *18*, 544–551. [CrossRef] [PubMed]
6. Prószéky, G. The (Hopefully Near) Future of Language Technologies. *Procedia Comput. Sci.* **2011**, *7*, 14–15. [CrossRef]
7. Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; Kuksa, P. Natural Language Processing (Almost) from Scratch. *J. Mach. Learn. Res.* **2011**, *12*, 2493–2537.
8. Young, T.; Hazarika, D.; Poria, S.; Cambria, E. Recent Trends in Deep Learning Based Natural Language Processing. *arXiv* **2017**, arXiv:1708.02709.

9. Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv* **2014**, arXiv:1406.1078.
10. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to Sequence Learning with Neural Networks. *arXiv* **2014**, arXiv:1409.3215.
11. Bahdanau, D.; Cho, K.; Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv* **2014**, arXiv:1409.0473.
12. Papineni, K.; Roukos, S.; Ward, T.; Zhu, W.J. BLEU: A method for automatic evaluation of machine translation. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics—ACL '02, Philadelphia, PA, USA, 7–12 July 2001; pp. 311–318.
13. Castilho, S.; Moorkens, J.; Gaspari, F.; Calixto, I.; Tinsley, J.; Way, A. Is Neural Machine Translation the New State of the Art? *Prague Bull. Math. Linguist.* **2017**, *108*, 109–120. [CrossRef]
14. Wu, Y.; Schuster, M.; Chen, Z.; Le, Q.V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv* **2016**, arXiv:1609.08144.
15. Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; Dauphin, Y.N. Convolutional Sequence to Sequence Learning. *arXiv* **2017**, arXiv:1705.03122.
16. Bergmanis, T.; Goldwater, S. Context Sensitive Neural Lemmatisation with Lematus. In Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, LA, USA, 1–6 June 2018.
17. Kann, K.; Schütze, H. Single-Model Encoder-Decoder with Explicit Morphological Representation for Reinflection. *arXiv* **2016**, arXiv:1606.00589.
18. Goldberg, Y. Neural Network Methods for Natural Language Processing. *Synth. Lect. Hum. Lang. Technol.* **2017**, *10*, 1–309. [CrossRef]
19. Groenewald, H.J. Educating Lia: The Development of a Linguistically Accurate Memory-Based Lemmatiser for Afrikaans. In *Intelligent Information Processing III*; Shi, Z., Shimohara, K., Feng, D., Eds.; Springer: Boston, MA, USA, 2007; Volume 228, pp. 431–440.
20. Mzamo, L.; Helberg, A.; Bosch, S. Introducing XGL—A lexicalised probabilistic graphical lemmatiser for isiXhosa. In Proceedings of the 2015 PRASA-RobMech International Conference, Port Elizabeth, South Africa, 26–27 November 2015; pp. 142–147.
21. van Huyssteen, G.B.; van Zaanen, M.M. Learning Compound Boundaries for Afrikaans Spelling Checking. In Proceedings of the First Workshop on International Proofing Tools and Language Technologies, Patras, Greece, 1–2 July 2004; pp. 101–108.
22. Pilon, S.; Puttkammer, M.J.; Van Huyssteen, G.B. Die ontwikkeling van'n woordafbreker en kompositumanaliseerder vir Afrikaans. *Literator* **2008**, *29*, 21–42. [CrossRef]
23. Schlünz, G.I.; Dlamini, N.; Kruger, R.P. Part-of-Speech Tagging and Chunking in Text-to-Speech Synthesis for South African Languages. In Proceedings of the INTERSPEECH 2016, San Francisco, CA, USA, 8–12 September 2016; pp. 3554–3558.
24. Du Toit, J.S. *A Comparative Evaluation of Open-Source Part-of-Speech Taggers for South African Languages*, 2017; Unpublished Project Report.
25. Eiselen, R. Government Domain Named Entity Recognition for South African Languages. In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC), Portorož, Slovenia, 23–28 May 2016; pp. 3344–3348.
26. Groenewald, H.J. Automatic Lemmatisation for Afrikaans. Master's Thesis, North West University, Potchefstroom, South Africa, 2006.
27. Groenewald, H.J. Using technology transfer to advance automatic lemmatization for Setswana. In Proceedings of the EACL 2009 Workshop on Language Technologies for African Languages, Athens, Greece, 31 March 2009; pp. 32–37.
28. Fick, M. Neurale netwerke as moontlike woordafkappingstegniek vir Afrikaans. *SA Tydskr. Nat. Tegnol.* **2003**, *22*, 2–5. [CrossRef]
29. Fick, M. 'n Masjienleerbenadering tot Woordafbreking in Afrikaans. Ph.D. Thesis, UNISA, Adelaide, Australia, 2013.

30. Li, H. Deep learning for natural language processing: Advantages and challenges. *Natl. Sci. Rev.* **2018**, *5*, 24–26. [CrossRef]
31. Deep Learning Approaches for Low-Resource NLP, DeepLo. Available online: https://sites.google.com/view/deeplo18/home (accessed on 14 August 2018).
32. Hedderich, M.A.; Klakow, D. Training a Neural Network in a Low-Resource Setting on Automatically Annotated Noisy Data. *arXiv* **2018**, arXiv:1807.00745.
33. Kann, K.; Bjerva, J.; Augenstein, I.; Plank, B.; Søgaard, A. Character-level Supervision for Low-resource POS Tagging. In Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP, Melbourne, Australia, 19 July 2018; pp. 1–11.
34. Goldberg, Y. A Primer on Neural Network Models for Natural Language Processing. *J. Artif. Intell. Res.* **2016**, *57*, 345–420. [CrossRef]
35. Bengio, Y.; Ducharme, R.; Vincent, P.; Jauvin, C. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.* **2003**, *3*, 1137–1155.
36. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, arXiv:1301.3781.
37. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; Dean, J. Distributed Representations of Words and Phrases and their Compositionality. *arXiv* **2013**, arXiv:1310.4546.
38. Sutskever, I.; Martens, J.; Hinton, G. Generating Text with Recurrent Neural Networks. In Proceedings of the 28th International Conference on Machine Learning, Bellevue, WA, USA, 28 June–2 July 2011; pp. 1017–1024.
39. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching Word Vectors with Subword Information. *arXiv* **2016**, arXiv:1607.04606. [CrossRef]
40. Nogueira dos Santos, C.; Zadrozny, B. Learning character-level representations for part-of-speech tagging. In Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 21–26 June 2014; Volume 32.
41. Pinter, Y.; Guthrie, R.; Eisenstein, J. Mimicking Word Embeddings using Subword RNN's. *arXiv* **2017**, arXiv:1707.06961.
42. Yu, X.; Faleńska, A.; Vu, N.T. A General-Purpose Tagger with Convolutional Neural Networks. *arXiv* **2017**, arXiv:1706.01723.
43. Wang, P.; Qian, Y.; Soong, F.K.; He, L.; Zhao, H. Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Recurrent Neural Network. *arXiv* **2015**, arXiv:1510.06168.
44. Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; Dyer, C. Neural Architectures for Named Entity Recognition. *arXiv* **2016**, arXiv:1603.01360.
45. Chiu, J.P.C.; Nichols, E. Named Entity Recognition with Bidirectional LSTM-CNN's. *arXiv* **2015**, arXiv:1511.08308.
46. Ma, X.; Hovy, E. End-to-end Sequence Labeling via Bi-directional LSTM-CNN's-CRF. *arXiv* **2016**, arXiv:1603.01354.
47. Dos Santos, C.N.; Guimarães, V. Boosting Named Entity Recognition with Neural Character Embeddings. *arXiv* **2015**, arXiv:1505.05008.
48. Zennaki, O.; Semmar, N.; Besacier, L. Unsupervised and Lightly Supervised Part-of-Speech Tagging Using Recurrent Neural Networks. In Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation (PACLIC), Shanghai, China, 30 October–1 November 2015; pp. 133–142.
49. Fang, M.; Cohn, T. Model Transfer for Tagging Low-resource Languages using a Bilingual Dictionary. *arXiv* **2017**, arXiv:1705.00424.
50. Fang, M.; Cohn, T. Learning when to trust distant supervision: An application to low-resource POS tagging using cross-lingual projection. *arXiv* **2016**, arXiv:1607.01133.
51. Plank, B.; Søgaard, A.; Goldberg, Y. Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models; Auxiliary Loss. *arXiv* **2016**, arXiv:1604.05529.
52. Faruqui, M.; Tsvetkov, Y.; Neubig, G.; Dyer, C. Morphological Inflection Generation Using Character Sequence to Sequence Learning. In Proceedings of the NAACL-HLT 2016, San Diego, CA, USA, 12–17 June 2016; pp. 634–643.

53. Schnober, C.; Eger, S.; Dinh, E.-L.D.; Gurevych, I. Still not there? Comparing Traditional Sequence-to-Sequence Models to Encoder-Decoder Neural Networks on Monotone String Translation Tasks. In Proceedings of the COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, Osaka, Japan, 11–16 December 2016; pp. 1703–1714.
54. Hellwig, O. Using Recurrent Neural Networks for joint compound splitting and Sandhi resolution in Sanskrit. In Proceedings of the 7th LTC, Poznań, Poland, 27–29 November 2015; pp. 289–293.
55. Dima, C.; Hinrichs, E. Automatic Noun Compound Interpretation using Deep Neural Networks and Word Embeddings. In Proceedings of the 11th International Conference on Computational Semantics, London, UK, 14–17 April 2015; pp. 173–183.
56. Pilon, S. 'n Woordsoortetiketteerder vir Afrikaans. *S. Afr. Linguist. Appl. Lang. Stud.* **2008**, *26*, 119–134. [CrossRef]
57. Taljard, E. A Comparison of Approaches to Word Class Tagging: Disjunctively vs. Conjunctively Written Bantu Languages. *Nord. J. Afr. Stud.* **2006**, *15*, 37.
58. Tjong, E.F.; Sang, K.; De Meulder, F. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003, Edmonton, AB, Canada, 31 May–1 June 2003; Volume 4, pp. 142–147.
59. CoNLL-U Format. 2017. Available online: http://universaldependencies.org/format.html (accessed on 2 August 2018).
60. Grave, E.; Bojanowski, P.; Gupta, P.; Joulin, A.; Mikolov, T. Learning Word Vectors for 157 Languages. *arXiv* **2018**, arXiv:1802.06893.
61. Wiseman, S.; Rush, A.M. Sequence-to-Sequence Learning as Beam-Search Optimization. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, TX, USA, 1–5 November 2016; pp. 1296–1306.
62. Gu, J.; Lu, Z.; Li, H.; Li, V.O.K. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, 7–12 August 2016; pp. 1631–1640.
63. Klein, G.; Kim, Y.; Deng, Y.; Senellart, J.; Rush, A.M. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *arXiv* **2017**, arXiv:1701.02810.
64. Britz, D.; Goldie, A.; M.-Luong, T.; Le, Q. Massive Exploration of Neural Machine Translation Architectures. *arXiv* **2017**, arXiv:1703.03906.
65. Bengio, S.; Vinyals, O.; Jaitly, N.; Shazeer, N. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. *arXiv* **2015**, arXiv:1506.03099.
66. Bohnet, B.; Nivre, J. A Transition-Based System for Joint Part-of-Speech Tagging and Labeled Non-Projective Dependency Parsing. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning Association for Computational Linguistics, Jeju Island, Korea, 12–14 July 2012; pp. 1455–1465.
67. University of Stuttgart. Mate Tools. Available online: http://www.ims.uni-stuttgart.de/forschung/ressourcen/werkzeuge/matetools.en.html (accessed on 16 August 2018).
68. Dorogush, A.V.; Ershov, V.; Gulin, A. CatBoost: Gradient boosting with categorical features support. *arXiv* **2017**, arXiv:1810.11363.
69. Al-Rfou, R.; Perozzi, B.; Skiena, S. Polyglot: Distributed Word Representations for Multilingual NLP. In Proceedings of the Seventeenth Conference on Computational Natural Language Learning, Sofia, Bulgaria, 8–9 August 2013; pp. 183–192.