

On the ability to bootstrap spellcheckers for Nguni languages via reusable model pipelines

Zola Mahlaza¹[0000–0001–9829–1480], C. Maria Keet²[0000–0002–8281–0853], and Tadiwa Magwenzi¹

¹ Department of Computer Science, University of Cape Town, South Africa
`zmahlaza@cs.uct.ac.za`, `MGWTAD001@myuct.ac.za`

² Computer Science Division, Stellenbosch University, South Africa
`cmkeet@sun.ac.za`

Abstract. Automated spellchecking is one of the core human language technologies. It is investigated intermittently for African languages, and then mostly for the most resourced among the low-resourced languages. Reusing spellchecker infrastructure to bootstrap spellcheckers for very low-resourced language remains an elusive goal. The aim of this work is to systematically investigate reusability of a pipeline to develop spellcheckers for all Nguni languages, and whether reusing a language model in one language for spellchecking in another may be feasible as an alternative bootstrapping option. We used the KenLM language model toolkit to train new models with texts in the four Nguni languages using n-grams (1 to 5), refined with a dynamic threshold. We then created a baseline with automatically wrecked words on their own language model, and assessed precision, recall, F1, and accuracy cross-lingually. Error detection accuracies are 86% for isiZulu, 83% for isiXhosa, 87% for isiNdebele, and 82% for siSwati. Average F1 score degradation is sometimes small (e.g., isiZulu text on isiNdebele model, isiXhosa on isiZulu), and sometimes substantial (siSwati on all three others). We therefore conclude that the approach may thus be feasible.

Keywords: Spellcheckers · Language Models · Agglutinating African Languages.

1 Introduction

The creation of basic natural language processing (NLP) and human language technologies (HLT) infrastructure for African languages may require the creation of reliable spellchecking tools, and spellchecking is generally considered one of the core tools, along with grammar checkers and morphological analysers. Work on spellchecking tools for South African languages dates back nearly 25 years, yet scant tooling is available and work on spellchecking for these and other African languages is largely fragmented. The most recent tooling made public are the Microsoft Office plugins by SADiLaR³, even though their technical details

³ <https://sadilar.org/en/spelling-checkers>

were neither published, thus negatively impacting the work’s reproducibility and replicability, nor was their accuracy reported. Earlier efforts predominantly used finite-state machines (e.g., [20]), wordlists for Open Office plugins⁴, a 3-gram based approach with a statistical word model [14, 15], and/or ensemble techniques [20]. For isiZulu, the statistical model reached up to 89% accuracy for error detection [15], error recall was 84% in [14], and the ensemble approach in [20] reached over 95% lexical recall, which is higher than reported in [14] but the former authors’ code or plugin is not available. Large language models (LLMs) have seen contemporary success across a variety of tasks, and several can now be used for South African language spellchecking (e.g., AfriqueLLM [2]). However, none of these models have been tested on their spellchecking accuracy, and they currently only support a subset of South African languages.

Regardless of the technique used, it would be beneficial for a cluster of languages or for a group of dialects if the same technique can be easily reused for languages other than the one it was designed for. For instance, Ndaba et al.’s algorithm to generate the 3-gram-based isiZulu model was shown to be reusable by feeding the implemented algorithm isiXhosa text to create an isiXhosa model [13]. This approach still requires corpus creation and cleaning, however, as well as setting thresholds. Alternatively, a language model-based spellchecker may perform sufficiently well across similar languages, not unlike improving neural machine translation by adding text from an adjacent language (e.g., [16]), data-driven noun class disambiguation across languages [5], and likewise for rule-based approaches of bootstrapping morphological analysers for the Nguni languages [3, 19] and bootstrapping controlled natural languages [6, 11]. For such cross-language use, the performance is expected to be less, but if it degrades from, say, 95 to 85% with respect to accuracy, then that 85% is still much better than no automated spellchecking.

We aim to investigate the ability to bootstrap spellcheckers across similar languages in this paper in a first experimental evaluation that demarcated to specifically the KenLM language model toolkit [9] and the four Nguni languages: isiZulu, isiXhosa, siSwati, and isiNdebele. We trained (created) spellchecking models for each language on a corpus for each language, and designed a novel spelling wrecker informed by the authors’ knowledge of typical typos (no data on common typos exist for the Nguni languages), which was applied to a large set of test words. Instead of hardcoded probability scores to determine whether each word is misspelled or not, we trained and use logistic regression models to dynamically learn the probability threshold(s) for identifying misspellings using the probability scores generated by the KenLM language models. We then tested the model pipelines for the error detection task on the wrecked words on their own and against the other three model pipelines.

Our results show that when each model is tested on its own language, the isiZulu and isiNdebele models performed best (F1 scores 0.75) while the siSwati model performed worst (F1 of 0.69). Zooming into the models’ performance for

⁴ Provided by translate.org.za in 2009 for Open Office versions predating v4: <https://extensions.openoffice.org/en/project/zulu-spell-checker.html>

each error type, it becomes evident that the models are very good at detecting some errors over others; e.g., the isiXhosa model pipeline has accuracies of 91% and 93% for insertion and transposition error types but lower values for deletion and phonetic errors. Regarding the core task of determining which language is best to use when bootstrapping, the results show that isiZulu is often the best model to rely on for the various Nguni languages.

The rest of the paper is structured as follows. We first describe in more detail the related work on spellchecking for African languages (Section 2). The main part of the paper is devoted to the materials and methods (Section 3) and results and discussion (Section 4). We conclude in Section 5.

2 Related work

There are a number of techniques that can be used for spellchecking to detect errors. For African languages specifically, typical strategies over the decades are to use morphological analysis with regular expressions [4, 8, 20], word lists and dictionaries (e.g., [22]), data-driven language models (statistical and possibly others as well) [14, 15], edit distance [17], a syllable-based approach [17], and ensemble techniques with one primary technique that is augmented with at least one other technique [7, 20, 14].

A key reason why not simply the software infrastructure of English has been reused, is the agglutinative nature of many African languages, as compared to the disjunctive orthography of English. Agglutination significantly reduces the type-to-token ratio, thereby rendering the wordlist-based approach impractical and unlikely to reach high accuracy rates. For instance, compare $\{ngibonga, ubonga, sibonga, ngiphuza, siphuza\}$ with their respective strings in English $\{‘I see’, ‘you see’, ‘we see’, ‘I drink’, ‘we drink’\}$ where for isiZulu there are 5 unique words (type-to-token of ratio of 1:1), whereas for English there are 5 types and 10 tokens (type-to-token ratio of 1:2). In addition, a downside for the wordlist approach is that the wordlists have to be collected for each language.

What the example also illustrates, is that there are recurring sequences in the words, thereby making resorting to a morphological analyser an appealing strategy, which was popular in the early 2000s [3, 4, 20]. For instance, the *ngi* is a recurring prefix, as are the *u* and *si* in the sample words above, and for simple present tense, the verb ends in *a*, to which one could add a list of verb roots (including *bong*, *phuz*). While one may expect it to reach higher accuracy than wordlists if all the rules are known and can be captured in regular expressions, the downside is that it is very labour intensive to develop, and it needs to be redone for each language. Similarity in rules vary, as has been demonstrated for isiZulu and isiXhosa verbs in detail by [11].

With the increase of digitisation and online publications, corpus-driven approaches to create language models (LMs) have become within reach. Ndaba et al. [15] investigated bi-, tri-, and quadri-grams based LMs, and trained the LMs with old and recent texts, observing that trigrams on recent text performed best. Their test data consisted of only $n = 46$ incorrect words. Their key idea is

to split up all strings in the corpus into strings of three consecutive characters (*ngi, gib, ibo, bon, ong, nga* etc.), compute their respective frequency, determine a cut-off threshold below which the strings are too unusual to likely be correct, and then it should be able to detect unseen strings and determine them as correct words or misspelled ones. A somewhat similar, more refined, approach that also aims to be resilient to the very long tail, was taken by Okgetheng et al. [17]: rather than n-grams, they used syllables. It was applied to the disjunctive Setswana, however, and we did not find a syllable detector, or else at least a list of confirmed syllables, for any of the Nguni languages.

Functioning tools for isiZulu spellchecking are sparse. To the best of our knowledge, only the SADiLaR Office365 plugins would be functional if one has paid the Office365 licence⁵, and there is Ndaba et al.’s [15] and related Mjaria and Keet’s [14] code and tool that are online available on GitHub and integrated in a platform-independent `jar` file that works with older Java versions⁶.

The lack of availability impedes reuse as well as conducting comparative performance experiments to standardise comparisons. Currently, the OpenOffice and Office365 plugins have no reported performance figures, some report accuracy [15], others report only lexical recall [20], or both lexical and error recall [14]. The same term/measure may also be calculated differently. For instance, Ndaba et al.’s accuracy is calculated as $\frac{TP+TN}{N}$ where TP denotes the true positives, instances where the model correctly predicted the positive word, TN denotes true negatives, cases where the model correctly identified the negative class, and N denotes the total number of words. On the other hand, Okgetheng et al.’s accuracy is calculated as the ratio of correct words to the total number of words [17].

Using the same infrastructure as Ndaba et al., Mashiane trained it with isiXhosa texts (corpus size and genre unknown) and obtained 79% accuracy, which presumably was calculated in the same way [13]. While not as good as the isiZulu spellchecker, it is, to the best of our knowledge, the only mention of reuse of infrastructure, where only text in the target language had to be fed into the algorithm.

3 Materials and Methods

This study evaluates how character-level language models for spellchecking perform for the languages they were created for and how they generalize to other languages. To this end, we designed two experiments. We first developed new models using corpora larger than in previous studies and generated realistic test datasets. We also evaluate cross-linguistic detection capabilities and employ data-driven methods to establish thresholds for error detection, moving beyond

⁵ The SADiLaR repository also lists and older isiNdebele spell checker [1], but the record has no files associated with it (last accessed on 15 March 2026).

⁶ The second version with the corrector is available at <https://github.com/normanpilusa/Spellchecker>; last accessed on 15 March 2026.

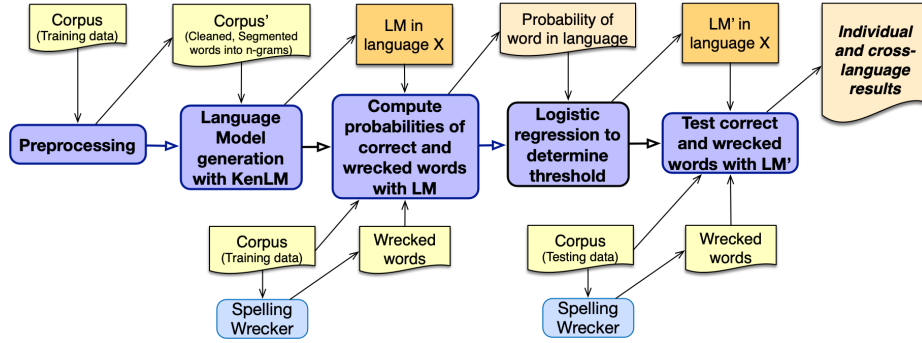


Fig. 1. Overview of the proposed pipeline for creating and evaluating the spellcheckers. Details of how each component is created or functions are described in the text.

the reliance on tailored and fixed values. The pipeline is summarised in Fig. 1 and described in the sections below.

The aim of Experiment 1 is to establish the complete training pipeline and test cases per language using a spelling wrecker tool with realistic error types, covering substitution, deletion, insertion, transposition, and phonetic errors. In this context, we consider a realistic error as a spelling mistake that a human would actually make when typing. As such, we do not test our spellcheckers on words where we randomly introduced errors to a word like *isikolo* ‘school’ but recognise that the a human is likely to produce the spelling error *siikolo* where the first two letters in the prefix are ordered incorrectly. We will describe the process of creating misspellings in the following sections.

Experiment 2 focuses on the creation of a cross-language matrix to determine the extent to which a model for one language can be used for another to serve as spellchecker.

3.1 Training and testing datasets

We first convert sentence-level corpora to obtain character-level corpora for model training in all Nguni languages: isiNdebele⁷, isiXhosa⁸, isiZulu⁹, and siSwati¹⁰. Corpora for all languages were sourced from SADiLaR. The datasets are processed in the following manner: removing the license statement from the original files, splitting the sentences by whitespace to identify the words, splitting all words that are combined using an “/” character, and generating a new

⁷ <http://rma.nwu.ac.za/index.php/resource-catalogue/isindebele-nchlt-text-corpora.html>

⁸ <http://rma.nwu.ac.za/index.php/resource-catalogue/isixhosa-nchlt-text-corpora.html>

⁹ <http://rma.nwu.ac.za/index.php/resource-catalogue/isizulu-nchlt-text-corpora.html>

¹⁰ <http://rma.nwu.ac.za/index.php/resource-catalogue/siswati-nchlt-text-corpora.html>

dataset where each line is made up of a single word where the characters are space separated. The resulting dataset is used to train new language models.

To move beyond fixed thresholds for determining whether the output of the language models is (in)correct, we developed a secondary dataset to train logistic regression models for each language. These classifiers use the output probabilities from our language-specific KenLM models to dynamically identify misspellings. To obtain a dataset for training such models, we rely on the Nguni language portion of the FineWeb [18] dataset. Specifically we use the training portion of the Lombard dataset¹¹. This process produced 144784 (isiNdebele), 187970 (isiXhosa), 226876 (isiZulu), and 265949 (siSwati) wrecked words that can be used to train logistic regression models to differentiate between correct and incorrect words, based on language model scores.

To wreck the words, we use a spelling wrecking script developed by the authors, which generates typos using five error types. The wrecker takes correct words from a corpus and then randomly applies one error per word. The substitution error replaces a character with a keyboard-adjacent key (e.g., ‘a’ into ‘s’, ‘q’, ‘w’, or ‘z’ based on QWERTY layout). Deletion removes a random character from the word and insertion adds either a keyboard-adjacent character or duplicates an existing character at a random position. The transposition error swaps two adjacent characters. The phonetic error type replaces characters with phonetically similar ones common in African languages ($b \leftrightarrow p$, $d \leftrightarrow t$, $g \leftrightarrow k$, $v \leftrightarrow f$, $z \leftrightarrow s$, $c \leftrightarrow k$).

We also create a small list of manually curated realistic spelling errors for testing the entire pipeline. Moreover to generate robust quantitative results, we use the test portion of the Lombard dataset. This process produced 2025 (isiNdebele), 210504 (isiXhosa), 222039 (isiZulu), and 7101 (siSwati) wrecked words.

3.2 Language models

We used the training dataset to create 5-gram language models (i.e., starting from unigrams up to 5-grams) using KenLM [9]. KeLM is a toolkit that implements a PROBING data structure using linear probing hash table and the TRIE that is “a trie with bit-level packing, sorted records, interpolation search, and optional quantization aimed at lower memory consumption”, together aimed a efficient language model queries both in time and memory [9]. It does not provide any pre-trained language models.

The newly created models are stored in the conventional ARPA format¹² where, after listing the number of n-grams in the range 1-5, each line specifies the token, its log probability, and the backoff weight to be used when accounting for unseen sequences of tokens. For instance, when using isiZulu for demonstrative purposes, the underlying language model has the following number of n-grams: unigrams = 122, bigrams = 3580 trigrams = 27779, 4 grams = 94909, and 5 grams

¹¹ <https://huggingface.co/datasets/anrilombard/sa-nguni-languages>

¹² http://www.seas.ucla.edu/spapl/weichu/htkbook/node243_mn.html

= 207476. It also has the following log probabilities and back-off weights, among many: (token = ‘F’, log probability = -1.986898, backoff weight = 0.60098976) and (token = ‘u’, log probability = -1.6911763, backoff weight = -1.0243318). Using the example from Section 2 on *ngibona*, the relevant entries include the following, among others: (-1.0848598 n g i -0.6555548) and (-0.43320304 n g i b o).

Each of the models can be used to generate a log probability for a word where the value is bounded above by zero. Consequently, words containing errors typically exhibit significantly larger negative magnitudes. For instance, the isiZulu word *Isahluko* ‘Chapter’ may have the log probability -5.3, which it has in our computed model, and all its misspellings would be expected to have even lower values, which is indeed the case for our model, including *isahbluko* (insertion error) has a log probability of -14.2 and *isahlumo* (substitution error) has as value -8.68.

Since one of our goals is not to rely on a fixed threshold regarding the probability when determining whether a word is misspelled, we also investigate techniques to dynamically detect errors and we discuss our approach in the next section.

3.3 Detecting errors via logistic regression

Fixed thresholds for determining whether a word is misspelled based on the output of the language models has the downside that it needs to be adjusted each for each language and for each training corpus, which is prohibitive for reuse. Therefore, we opted to devise a way to create models that can dynamically identify the threshold. Specifically, we developed supervised binary classification models, implemented via PyTorch¹³ to learn how to determine whether a model is misspelled based on the log probabilities produced by the language models. The new models are designed to map the scores obtained from each raw character-level language model to a probability of correctness. We train the models using the training portion of the Lombard dataset, annotated with scores computed by the KenLM language models, to create a dataset where “correct” word scores are labelled as positive samples and wrecked word scores are labelled as negative samples. We train new models for each of the languages using stochastic gradient descent and a binary cross-entropy loss function to refine the model’s weights. Our configuration uses a learning rate of 0.01, a batch size of 16, and a total number of 5 epochs per language.

3.4 Experiment 1: Baseline testing

The goal of this experiment is to measure the extent to which each language model, and associated logistic regression model, can differentiate between correct and incorrect words per language. We conducted this investigation by feeding the 2025 (isiNdebele), 210504 (isiXhosa), 222039 (isiZulu), and 7101 (siSwati)

¹³ <https://pytorch.org/>

wrecked words to their respective language models and computed the likelihood score, and used the appropriate logistic regression model to ascertain whether the models are (in)correct. When then established the performance of the model pipelines using the precision scores (i.e., $\frac{TP}{TP+FP}$), recall (i.e., $\frac{TP}{TP+FN}$), and F1 (i.e., $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$) for both the correct and incorrect words, summarised using the usual notion of a confusion matrix where the formulae acronyms have the following meanings: TP = true positives (correctly spelled and predicted correct), FP = false positives (wrongly spelled but predicted correct), FN = false negatives (correctly spelled but predicted incorrect), and TN = true negatives (wrongly spelled and predicted wrong). We also calculated the error accuracy, i.e., the percentage of misspelled words that are correctly identified for each error type.

3.5 Experiment 2: cross-model testing

The second experiment is designed to determine whether a spellchecker for one language can be effectively applied to a related language, as opposed to one that is more distantly related. For example, given their closer linguistic proximity and shared Zunda features according to the Guthrie classification (see [12]), an isiZulu spellchecker may be better suited for isiXhosa than an siSwati spellchecker would be. Consequently, this experiment evaluates the performance variance of language tools within the Nguni group when used cross-linguistically. For this experiment, we use the same dataset as experiment 1. We also compute the precision, recall, and F1 scores for the (in)correct words, in the same way as described in the previous section.

4 Results and Discussion

We first describe the quantitative results for same-language assessment, functioning as the baseline, and then for cross-language assessment. We then perform a qualitative analysis in Subsection 4.3, focussing on elucidating the main cause(s) of not reaching 100% accuracy.

4.1 LM performance on the same language

The performance of the models for each language is provided in Table 1. When analysing the performance of the models, it can be established that the combination of the language and logistic regression models obtains an average F1 score of 0.74 for incorrect words and 0.72 for correct words across the languages. While all models are able to identify misspellings, to a reasonable extent, the siSwati model exhibits lower performance when classifying incorrect words. This pattern can also be seen when averaging the F1 scores for the (in)correct words where siSwati has the lowest score (i.e., 0.69).

We calculated the accuracies of the model pipelines for each of the error types and the results are given in Table 2. When focusing on the isiXhosa pipeline,

Table 1. Metrics for Nguni Language spellcheckers for each language. Wrecked words are labelled incorrect words and words without spelling errors are labelled correct words.

Language	Incorrect Words			Correct Words			Avg F1
	Prec	Rec	F1	Prec	Rec	F1	
isiNdebele	0.724	0.797	0.759	0.774	0.696	0.733	0.746
isiXhosa	0.724	0.769	0.746	0.753	0.707	0.729	0.7375
siSwati	0.664	0.773	0.714	0.728	0.609	0.663	0.6885
isiZulu	0.730	0.792	0.759	0.772	0.707	0.738	0.7485

we see that the model performs best for insertion and transposition where the pipeline obtains accuracies of 91% and 93% respectively. This pattern can be observed for all languages where the pipelines perform best for transposition and insertion error types.

Table 2. Percentage of correctly identified misspellings per error type

Error Type	IsiZulu	isiXhosa	IsiNdebele	siSwati
Substitution	0.8944	0.8796	0.899	0.8256
Deletion	0.6439	0.6105	0.6786	0.609
Insertion	0.9385	0.912	0.9497	0.9267
Transposition	0.9491	0.9295	0.9438	0.9143
Phonetic	0.5348	0.5138	0.5302	0.5925

4.2 Cross-lingual performance of the LMs

The cross-lingual performance results of the models are provided in Table 3. The results indicate that, while all models successfully identify misspellings, the geographical proximity between the Nguni group dictates cross-linguistic performance. In particular, the siSwati model pipeline exhibits lower accuracy when classifying incorrect words compared to the Zunda cluster (i.e., isiZulu, isiXhosa, and isiNdebele), which shows greater mutual performance. The greatest F1 score 0.684 for siSwati is noticeably lower than the greatest scores for isiNdebele, isiXhosa, and isiZulu. The same pattern can be observed for the correct words.

For isiNdebele, isiXhosa, and isiZulu, the best performing model pipeline, by F1 scores, is consistent across correct and incorrect words. For example, when bootstrapping a model pipeline for isiNdebele, it is best to rely on an isiZulu model pipeline instead of the other languages. The second best choice is to rely on an isiXhosa model pipeline. In the case of siSwati, however, the best performing pipeline to rely on when processing incorrect words is different

from the best pipeline for correct words. This is demonstrated by the fact that when processing incorrect words, it is best to rely on an isiNdebele model and for correct words, it is better to rely on an isiZulu model. This implies that depending on the chosen language model, there may be more false positives for the two categories of words.

Table 3. Cross language performance metrics for Nguni spellcheckers. Highest scores per language are highlighted in bold.

Language → Model	Incorrect Words			Correct Words			Avg F1
	Prec	Rec	F1	Prec	Rec	F1	
isiNdebele → isiXhosa	0.627	0.827	0.713	0.746	0.509	0.605	0.659
isiNdebele → isiZulu	0.649	0.814	0.722	0.751	0.559	0.641	0.6815
isiNdebele → siSwati	0.591	0.867	0.703	0.750	0.400	0.522	0.6125
isiXhosa → isiNdebele	0.621	0.844	0.716	0.757	0.485	0.591	0.6535
isiXhosa → isiZulu	0.654	0.810	0.724	0.751	0.573	0.650	0.687
isiXhosa → siSwati	0.576	0.873	0.694	0.738	0.357	0.481	0.5875
isiZulu → isiNdebele	0.649	0.847	0.735	0.780	0.542	0.640	0.6875
isiZulu → isiXhosa	0.642	0.834	0.726	0.764	0.536	0.630	0.678
isiZulu → siSwati	0.602	0.876	0.714	0.772	0.422	0.546	0.63
siSwati → isiNdebele	0.582	0.829	0.684	0.702	0.404	0.513	0.5985
siSwati → isiXhosa	0.577	0.821	0.678	0.690	0.398	0.505	0.5915
siSwati → isiZulu	0.592	0.801	0.680	0.692	0.447	0.543	0.6115

4.3 Qualitative analysis of false positives and negatives

Inspecting the false negatives of the incorrectly spelled words for the isiXhosa dataset when using the isiZulu model, we observe a number of issues. First, the wrecker does not work well for phonetic wrecking, where relatively many ‘wrecked’ words are the same string as the correct words (being 57% for isiNdebele, 53% for isiXhosa, 51% for isiZulu, and 51% for siSwati), and therefore the checker evaluates ‘false’, i.e., determines (correctly) that it is a correctly spelled word rather than the incorrect expected. Second, a good number of deletions have a character deleted such that the wrecking has resulted in a different valid word rather than a non-word. For instance, *indlela* ‘path’ (a.o.t.) is wrecked to *idlela* ‘he/she/it eats for’, *bonke* ‘all’ (noun classes 2, 2a, 14) into *onke* ‘all’ (noun class 6), and *futhi* ‘and’ and *uthi* ‘stick’ (a.o.t.). This issue appears much less of a problem with the other three types of errors.

There are, of course, true negatives, such as *kuth* being incorrectly flagged as not incorrect, whereas it should have been easy to detect it as incorrect because very few words end in *th*. The latter holds likewise for, e.g., *futh* in the isiZulu set itself. The spellchecker of Mjaria and Keet [14] (the `Spellchecker.jar`

file in the .zip download from https://people.cs.uct.ac.za/~mkeet/files/Spellchecker_final2.zip) also does not detect these words as having an error.¹⁴

We also observed a notable difference in word lengths regarding the model’s ability to detect spelling errors. We analysed the lengths of the wrecked words, categorized by whether the pipeline predicted them as correct or incorrect; the resulting average lengths are reported in Table 4. Our findings suggest the model performs effectively on longer words but struggles to accurately identify errors in shorter ones.

Other language features, such as tonality [10], cannot be investigated with this corpus-based language model at present, because it is not reflected in the orthography of the Nguni languages. Investigating the effect of the corpus on the F1 and accuracy scores is possible, but already know to affect the spellchecking performance as shown by [15]. Future improvements in spellchecking performance for this pipeline might be obtained with other LM generation algorithms than KenLM, creating lists of syllables and include it in the training set, or similar strategies.

5 Conclusions

The newly created pipeline for spellchecker generation with KenLM, a new wrecker, and logistic regression for dynamic threshold setting performed well on error detection accuracy for words in the language of the language model, scoring 86% for isiZulu (average F1: 0.75), 83% for isiXhosa (average F1: 0.74), 87% for isiNdebele (average F1: 0.75), and 82% for siSwati (average F1: 0.69). Cross-lingual spellchecking saw an average F1 score degradation that can be small (among others, isiZulu text on the isiNdebele language model, isiXhosa on isiZulu), and sometimes relatively substantial (siSwati on all three others, down to around 0.6 average F1). In all cases, the F1 score is higher for detecting incorrect words than for detecting correct words. We therefore conclude that the bootstrapping and cross-lingual use approach can be feasible.

Future work involves controlled testing with other (very) low resourced languages in what are by various measures deemed close and distant languages (e.g., [21]) to gain a deeper understanding of the potential, improve the accuracy, and proceed to new tasks and questions, such as determining actual common misspellings in the Nguni languages. Also, we plan to collect more datasets to investigate the problem in cases where the languages have similar dataset sizes.

Supplemental material. The materials have been anonymised for review and can be downloaded from <https://github.com/AdeebNqo/bootstrap-spellcheckers>.

Acknowledgments. This work was financially supported in part by the National Research Foundation (NRF) of South Africa (Grant Number and CPRR23040389063).

¹⁴ The authors could not test them with the SADiLaR plugin, because none of the authors have Windows to run the .exe nor a Virtual Machine with a compatible MS Office installed.

Table 4. Number of words by error type, and average word lengths for incorrect words predicted to be correct and incorrect (Values rounded off).

		Pred. as correct	Predicted as incorrect
Error Type	Number of words	Avg Length	Avg Length
IsiNdebele			
Deletion	392	8	6
Insertion	398	9	7
Phonetic	430	9	7
Substitution	396	9	6
Transposition	409	8	6
IsiZulu			
Deletion	44562	8	6
Insertion	44423	9	7
Phonetic	44497	9	7
Substitution	44290	9	6
Transposition	44267	8	6
IsiXhosa			
Deletion	42162	8	5
Insertion	42296	9	7
Phonetic	42355	9	7
Substitution	41789	8	6
Transposition	41902	8	6
SiSwati			
Deletion	1445	8	5
Insertion	1418	9	6
Phonetic	1416	9	7
Substitution	1422	9	6
Transposition	1400	8	5

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. isiNdebele spelling checker 1.0 (2013), <https://hdl.handle.net/20.500.12185/186>, SADIaR Language Resource Repository
2. Ajayi, O.A.: What do prompts reveal about model capabilities in low-resource languages? In: 7th Workshop on African Natural Language Processing (2026), <https://openreview.net/forum?id=7JZmTp85Yf>
3. Bosch, S., Pretorius, L., Fleisch, A.: Experimental bootstrapping of morphological analysers for Nguni languages. *Nordic Journal of African Studies* **17**(2), 23 (Jun 2008). <https://doi.org/10.53228/njas.v17i2.237>
4. Bosch, S.E., Eisele, R.: The effectiveness of morphological rules for an isiZulu spelling checker. *South African Journal of African Languages* **25**(1), 25–36 (2005)
5. Byamugisha, J.: Noun class disambiguation in Runyankore and related languages. In: *Proceedings of the 29th International Conference on Computational Linguistics*.

- pp. 4350–4359. International Committee on Computational Linguistics, Gyeongju, Republic of Korea (Oct 2022), <https://aclanthology.org/2022.coling-1.383>
6. Byamugisha, J., Keet, C.M., DeRenzi, B.: Bootstrapping a Runyankore CNL from an isiZulu CNL. In: Davis, B., et al. (eds.) 5th Workshop on Controlled Natural Language (CNL'16). LNAI, vol. 9767, pp. 25–36. Springer (2016), 25–27 July 2016, Aberdeen, UK.
 7. Cissé, T.I., Sadat, F.: Automatic spell checker and correction for under-represented spoken languages: Case study on Wolof. In: Mabuya, R., Mthobela, D., Setaka, M., Van Zaanen, M. (eds.) Proceedings of the Fourth workshop on Resources for African Indigenous Languages (RAIL 2023). pp. 1–10. Association for Computational Linguistics, Dubrovnik, Croatia (2023). <https://doi.org/10.18653/v1/2023.rail-1.1>
 8. Grobbelaar, L.A., Kinyua, J.D.M.: A spell checker and corrector for the native South African language, South Sotho. In: Proceedings of the 2009 Annual Conference of the Southern African Computer Lecturers' Association. p. 50–59. SACLA '09, Association for Computing Machinery, New York, NY, USA (2009). <https://doi.org/10.1145/1562741.1562747>
 9. Heafield, K.: KenLM: Faster and smaller language model queries. In: Callison-Burch, C., Koehn, P., Monz, C., Zaidan, O. (eds.) Proceedings of the Sixth Workshop on Statistical Machine Translation, WMT@EMNLP 2011, Edinburgh, Scotland, UK, July 30–31, 2011. pp. 187–197. Association for Computational Linguistics (2011), <https://aclanthology.org/W11-2123/>
 10. Maddieson, I., Sands, B.: The sounds of the bantu languages. In: de Velde, M.V., Bostoen, K., Nurse, D., Philippson, G. (eds.) The Bantu Languages, chap. 2, pp. 79–127. Routledge, 2 edn. (2019)
 11. Mahlaza, Z., Keet, C.M.: A method for measuring verb similarity for two closely related languages with application to Zulu and Xhosa. South African Computer Journal **31**(2), 34–56 (2019)
 12. Maho, J.: A classification of the Bantu languages: An update of Guthrie's referential system. In: Nurse, D., Philippson, G. (eds.) The Bantu Languages, chap. 7, pp. 639–651. Routledge (2014)
 13. Mashiane, N.: Data-driven statistical model based error detector for IsiXhosa. Master's thesis, Department of Computer Science, University of Cape Town, South Africa (2016), https://projects.cs.uct.ac.za/honsproj/cgi-bin/view/2017/mashiane_mjaria_net_i.zip/documents/NthabisengMashiane.pdf, last accessed: 14 March 2026
 14. Mjaria, F., Keet, C.M.: A statistical approach to error correction for isiZulu spellcheckers. In: Cunningham, P., Cunningham, M. (eds.) IST-Africa 2018. p. 9p. IIMC International Information Management Corporation (2018), 9–11 May, Botswana
 15. Ndaba, B., Suleman, H., Keet, C.M., Khumalo, L.: The effects of a corpus on isiZulu spellcheckers based on n-grams. In: Cunningham, P., Cunningham, M. (eds.) IST-Africa 2016. IIMC International Information Management Corporation (2016), 11–13 May, 2016, Durban, South Africa
 16. Nyoni, E., Bassett, B.A.: Low-resource neural machine translation for southern African languages. arXiv preprint arXiv:2104.00366 (2021), <https://arxiv.org/pdf/2104.00366.pdf>
 17. Okgetheng, B., Malema, G., Ahmer, A., Lenyibi, B., Ishmae, O.: Bantu spell checker and corrector using modified edit distance algorithm (MEDA). In: Wyld, D.C., et al. (eds.) ARIA, SIPR, SOFEA, CSEN, DSML, NLP, EDTECH, NCWC - 2022. pp. 303–316. AIRCC (2022). <https://doi.org/10.5121/csit.2022.121524>

18. Penedo, G., Kydlíček, H., Sabolcec, V., Messmer, B., Foroutan, N., Kargaran, A.H., Raffel, C., Jaggi, M., von Werra, L., Wolf, T.: Fineweb2: One pipeline to scale them all - adapting pre-training data processing to every language. *CoRR abs/2506.20920* (2025). <https://doi.org/10.48550/ARXIV.2506.20920>, <https://doi.org/10.48550/arXiv.2506.20920>
19. Pretorius, L., Bosch, S.E.: Exploiting cross-linguistic similarities in zulu and xhosa computational morphology: Facing the challenge of a disjunctive orthography. In: *Proceedings of the EACL 2009 Workshop on Language Technologies for African Languages - AfLaT 2009*. pp. 96–103 (2009)
20. Prinsloo, D.J., de Schryver, G.M.: Spellcheckers for the South African languages, part 2: the utilisation of clusters of circumfixes. *South African Journal of African Languages* **8**, 83–94 (2004)
21. Rexová, K., Bastin, Y., Frynta, D.: Cladistic analysis of bantu languages: a new tree based on combined lexical and grammatical data. *Naturwissenschaften* **93**(3), 189–194 (2006). <https://doi.org/10.1007/s00114-006-0088-z>
22. Salifou, L., Naroua, H.: Design of a spell corrector for Hausa language. *international Journal of computational Linguistics (IJCL)* **5**(2), 14–26 (2014)