# A statistical approach to error correction for isiZulu spellcheckers

Frida MJARIA[1], C. Maria KEET[1]
[1]*University of Cape Town, Cape Town, 7701, South Africa*
*Tel: +27 (021) 650 2667, Fax: +27 (021) 650 3557, Email: fmjaria@gmail.com, mkeet@cs.uct.ac.za*

**Abstract:** Spellcheckers have become important due to the increase of text-based communication at work and in society on social media. There is, however, very little support for spellchecking in agglutinating Sub-Saharan African (Bantu) languages. While error detection has shown to yield acceptable results for at least isiZulu, error correction has not even been investigated. The aim of this paper is to solve the spelling correction problem by means of a statistical approach such that it can provide candidate corrections to misspelled isiZulu words (non-word errors). Trigrams learned from a corpus, their probabilities, minimum edit distance, and additional optimisations are used in the error corrector. The corrector was evaluated for the four types of non-word errors (substitution, insertions, deletions, and transpositions). It achieved an 89% language recall rate, 84% error recall, 85% language precision, and 88% error precision for error correction. The error corrector was found to have an overall suggestions accuracy rate of 95% and relevance of 61%, performing best for transposition errors. The error corrector has been added to an existing open source isiZulu error detector. This facilitates uptake and, moreover, fills a feature gap that has numerous benefits for society, both for isiZulu speakers and learners, and for bootstrapping spellcheckers for related languages.

**Keywords:** spellchecker, non-word error correction, statistical language model, isiZulu.

## 1. Introduction

A large amount of text is generated online on a daily basis that is informal and unedited by nature, requiring spelling error detection and correction [4,9,13] over and above spellcheckers' common use in the workplace. Spellcheckers aim to detect two types of spelling errors, namely, non-word and real word (context-based) errors. Non-words errors are words that do not occur in a given language. These errors are usually caused by typographical errors made by the user or by spelling a given word according to its pronunciation, which we focus on in this paper. While spellcheckers exist for globally dominant languages, there is scant spellchecker support for Sub-Saharan indigenous languages notwithstanding substantial language use. We zoom in on the most widely spoken language in South Africa, isiZulu, as 22.7% of the population have it as a first/home language and another about 23% can speak it to varying degree[1]. Only two isiZulu spellcheckers are currently partially functional. Ndaba et al.'s spellchecker uses a data-driven statistical model and n-grams to detect misspelled words, is reported to have an accuracy of 89% [11], and its technology has been incorporated in a production-level spellchecker[2]. Spellchecker.net[3] can detect and correct misspelled words, but the

[1] Nguni on Britannica Academic: 2017. http://academic.eb.com/levels/collegiate/article/55655. Accessed: 2017- 05- 08.
[2] http://ulpdo.ukzn.ac.za/HLTechnologies/SpellChecker.aspx

technology and its accuracy is not disclosed, it requires an active Internet connection, and is clogged with advertisements.

The aim of this paper is to fill this gap in isiZulu spellchecking by developing an error corrector that can accurately provide candidate corrections to incorrectly spelled isiZulu words in text. Given that a statistical approach was shown to be feasible for detection [11], we pursued a statistical approach for error correction. First, a language model was built with a corpus from which trigrams and their probabilities were constructed. Second, to achieve error correction, the model was augmented with the minimum edit distance algorithm, a new successive trigram probabilities check, a novel optimisation for transpositions, and further exploitation of trigram probabilities for ordering candidate corrections. This was evaluated on accuracy and relevance. For the test set, it obtained language and error recall and precision around 80-90%, with accuracy (suggesting something) and relevance (suggestion set contains intended word) fluctuating by type of spelling mistake. Finally, to make the step toward technology impact in business and society, the error corrector has been integrated with an optimised version of Ndaba et al.'s error detector core to form a fully functional isiZulu spellchecker.

The paper consists of the following sections: related work, system design and implementation that describes the core technological novelties, experiment design and results, discussion, and conclusions and future works.

## 1. Related Work

A spellchecker consists of a body of text representing the language (corpus), an Error model (EM) and a Language model (LM) [12]. An LM is used to determine how frequent a word occurs in a dictionary. A corpus can be used to build an LM [6]. An EM is an algorithm used for modelling spelling errors [12], whose accuracy is affected by the corpus used [11,3,5,18]. The LM can also be affected by the corpus so that candidate corrections may receive an inaccurate higher or lower LM score [11,12,18]. Such text-dependent EMs and LMs typically also use n-grams in addition to wordlists. An n-gram is an n-letter subsequence of words or a string, where n is usually one, two, or three and sometimes four [9]; e.g., for *kodwa*, the trigrams are *kod*, *odw*, and *dwa*. Such character n-grams can then be used as a dictionary of the spellchecker [7], which might improve the number of input words that a spellchecker identifies as correctly spelled [3,7]: instead of checking a word in a dictionary, each n-gram in the input word is compared with n-grams stored in the dictionary. If any n-gram in an input word is not found or its occurrence in the corpus is below a set frequency threshold, the word is flagged as misspelled [11]. The use of n-grams has also been used for error correction (e.g., [1,3,5,13]). This is then augmented with typically at least the Levenshtein distance (or its variation)[4] for error correction [3,13,18], but also with, among others, confidence classifiers constructed using the noisy channel model [18] and a Bayesian approach [5].

Once candidate corrections have been computed from the detected errors, they have to be ranked [10]. The Levenshtein distance assists in that candidate corrections with higher edit operations are less favoured than candidate corrections with lower edit operations [6, 10]. Further, a language model score can be used to select the best candidate correction(s). For instance, by assigning a score to each n-gram according to the frequency or the likelihood of that n-gram occurring in the corpus so that words that are more frequently used in the language will be favoured as the best candidate corrections as opposed to words

[3] https://www.spellchecker.net/africa_zulu_spell_checker.html

[4] The Damerau-Levenshtein distance (DL), also known as the minimum edit distance, is the staple algorithm used to calculate the minimum number of changes required to transform one word into another by means of insertion, deletion, substitution and/or transposition operations.

that are less commonly used. This also helps with candidate words that may have the same edit distance away from the misspelled word [6,10,12].

Regarding types of spelling errors, it is claimed that 80% of spelling errors occur from a single substitution, insertion, deletion, or transposition error in a word [17], which are the four types of non-word errors [13]. Substitution is when a letter in a word is replaced with another, insertion is when a letter is added to a word, deletion is when a letter is omitted from a word, and transposition is when a swap occurs between two adjacent letters in a given word [1,6]. While non-word errors are categorized as non-existent words in a language, real-word errors denote words that are correctly spelled, but are used in the wrong context within a sentence [12,15]. The latter requires automatic sentence analysis, which is beyond reach for isiZulu at present.

## 2. System design and implementation

This section details the design and implementation of the techniques used to construct the error corrector. The error corrector focuses on non-word errors originating from a one-character change from the intended word. The design of the error corrector was implemented using the Java programming language, as it is a platform independent, portable, object-oriented programming language that offers vast libraries to ease development, and Java was used to develop the error detector.

### *2.1 Preliminaries*

The corpus used was the same as for the optimised error detector of [11] (fn. 2), being a section of the isiZulu National Corpus [8], which comprises isiZulu articles and novels. The corpus was cleaned by removing non-isiZulu words and punctuation (excluding apostrophes and hyphens occurring in isiZulu). The text files were then used to construct trigrams of each word in the corpus, from which a unique trigram list was constructed to calculate their frequency and various probabilities described below.

### *2.2 Corrector algorithms*

When a word from the input text is flagged as incorrect by the error detector, the word is broken up into its trigram constituents. The list of unique trigrams—stored in a HashMap and an ArrayList for fast traversal—is traversed to identify every incorrect trigram in the word. This list contains only trigrams with frequencies greater than the frequency threshold to improve the system's performance.

A trigram is deemed incorrect if it does not appear in the trigram list or if it occurs in the list but its frequency is below the assigned frequency threshold. Once a trigram is identified as incorrect, the Levenshtein distance of the trigram and each unique trigram is calculated. All the trigrams that yield an edit distance of two or less are stored in an array. This process is repeated for each trigram that is identified as incorrect.

Subsequently, the ArrayList containing zero or more candidate trigrams is stored together with the incorrect trigram. This is achieved by creating a Trigram object for each trigram of the incorrect word, which consists of an ArrayList to store candidate corrections and a String variable to store the incorrect trigram.

Once all the incorrect trigrams are identified and candidate trigrams are computed, the trigrams are joined to form possible isiZulu words to display as candidate corrections to the user. The candidate corrections are formed using a brute force algorithm to generate the possible combinations from the stored candidate trigrams, including combinations with the identified as correctly spelt trigrams of the flagged word. Due to the computational intensity of the algorithm, the ArrayLists containing candidate trigrams are ordered and the

BinarySearch Algorithm is used to determine the start and end indices to perform combinations on. Due to the agglutinative nature of isiZulu, some of the candidate corrections obtained from this algorithm might be non-existent in the isiZulu language. To mitigate this, a list of unique words from the corpus is traversed (stored in a HashSet for quick access). All candidate corrections that do not occur in this list are discarded.

Finally, in order to show the most relevant suggestions to users, candidate corrections are ordered according to their probability score as follows. Each candidate correction is broken up into its trigram constituents and stored in a ProbScore object in an ArrayList. For each trigram (excluding the trigram occurring at index 0), the probability of the trigram occurring after the previous trigram is obtained. The probabilities are summed up and divided by the total number of trigrams. The candidate corrections are then sorted according to their probability score and at most 10 suggestions are displayed to the user.

### *2.3 Optimisations*

Two optimisations were added. The first, and novel, technique is to check for a transposition error. This is done by checking if the trigram contains two adjacent consonants. If it does, the consonants are swapped and the HashMap containing the unique trigram is checked to see if that trigram exists. If it does, string manipulation is used to formulate a new word containing the trigram with the swapped adjacent consonants. The HashSet containing unique words is checked to see if the new word is a correct isiZulu word. If it is correct, the word is displayed as a candidate correction and the suggestions algorithm is terminated for this word.

Probabilities are used in the second method to improve on the accuracy of the error corrector in finding candidate corrections. Due to the agglutinative nature of isiZulu, a trigram can be viewed as correct by the error detector, but is incorrect in the given word. When obtaining candidate corrections through string manipulation, the correct trigram is joined to the candidate trigram to form a new string. If the trigram occurring before the incorrect trigram does not belong in the word, and the trigram is recognized as correct, the intended word will not appear as a candidate correction. To mitigate this, the probability of a trigram occurring after a specific trigram is calculated for all combinations of unique trigrams that have a frequency >= 45 that can be combined to form a new string. These probabilities are stored in a text file and accessed as needed.

## 3. Experiment design

The aim of the experiment is to determine the performance of the spellchecker.

### *3.1 Dataset*

The INC section provided by Dr. Khumalo was used for training the system and an isiZulu wordlist of the released 2016 isiZulu spellchecker (detector-only)[5] was used as the testing dataset to test that released spellchecker (fn. 3), assuming that each word in the wordlist is correctly spelled. Instead of Ndaba's approach of introducing a few typos to compute accuracy, this is scaled up:.6000 words from the wordlist are fed into an error generating module[6] to generate non-word spelling errors, 1500 for each type of single-character error. This method is used to create spelling errors as no resource containing common misspellings occurring in isiZulu could be found.

[5] https://github.com/normanpilusa/Isizulu_Spellchecker/blob/master/correct.txt
[6] https://github.com/benjholla/spellwrecker/blob/master/SpellWrecker/src/spellwrecker/components/spellwreckers/QwertySpellWrecker.java

### *3.2 Evaluation metrics*

A confusion matrix is used to classify the results into 4 categories: True Positives (TP), True Negatives (TN), False Negatives (FN) and False Positives (FP). TP denotes the number of words that are known to be correctly spelled that the system recognizes as correctly spelled. TN is the number of incorrectly spelled words that the system flags as incorrect. FP denotes the number of incorrectly spelled words that the system flags as correctly spelled. FN is the number of correctly spelled words that the system recognizes as incorrect [11]. Recall denotes how representative the lexicon of the spellchecker is of the language and how free the lexicon used of incorrectly spelled words is [19]. The *Lexical Recall* (LR) [19] is calculated as LR = TP/(TP+FN). To determine how error-free the corpus is, we calculate the *Error Recall* (ER) of the spellchecker [19] as ER = TN/(TN+FP). Precision measures denote how accurate the spellchecker is at recognizing correctly spelled words (*lexical precision* (LP)) and flagging incorrectly spelled words (*error precision* (EP)) [19]. LP and EP calculated using the following equations: LP = TP/(TP+FP) and EP = TN/(FN+TN). Suggestion Adequacy (SA) denotes the ability of the spellchecker to provide accurate suggestions that are relevant to the user [19]. *Accuracy* (*Cs*) is the number of TN that the error model was able to provide candidate corrections for. This, however, does not determine the *relevance* (*Cv*) of the suggestions. *Cv* denotes the number of corrections provided for TN that contain the intended word that was incorrectly spelled. To determine *Cv*, a scoring system is used based on the combination of scoring systems used by [12,20]: if the suggestions contain the intended word, *Cv* = 1; if the suggestions do not contain the word, *Cv* = 0.5; if no suggestions provided, *Cv* = 0. This scoring system is performed on each word and the total sum for all TN determines *Cv*. To obtain the accuracy and relevance percentage, *Cv* and *Cs* are divided by TN.

### *3.3 Performance experiments*

Since the corpus utilized affects a spellchecker's performance, tests were carried out to determine the threshold[7] (details omitted due to space limitations) and error detection recall. Recall was calculated in two different ways: 1) over the 2016 release (fn. 2) and newly-built model with a clean corpus using a list of 12453 words assumed to be correct and 2) at the end with the 2016 release, this new version, and Spellchecker.net using small sections of text (due to tool restrictions) from Ukwabelana [16], the INC [8], and news items [11].

To determine the error correction performance for each type of spelling error, each 1500-word file containing the spelling errors was divided into three sections of 500 words to ascertain that the system provides consistent results for each type of spelling error. Each section was fed to the system and the resulting TN, FP, *Cs* and *Cv* values were stored. This was repeated for each type of spelling error. Finally, the system was tested using the testing dataset with correctly spelled words. The resulting TP and FP values were used together with the cumulative TN and FN values to generate a confusion matrix. The confusion matrix together with the cumulative *Cv* and *Cs* values were used to measure the recall, precision, accuracy and SA of the system to determine the accuracy of the spellchecker.

## 4. Results

Calibration of the system[8] showed a threshold of 45 for trigram frequency to be the best trade-off between FN and FP, which amounts to the top 2199 most frequent trigrams, which is slightly lower than Ndaba et al.'s number of top-trigrams (2569). Error detection recall

[7] Ndaba et al.'s threshold of 0.003 could not be used, as it was miscalculated over the number of unique trigrams rather than total number of trigrams, and we cleaned the corpus.
[8] Tool, source code, and data: http://www.meteck.org/files/spellchecker/.

for the 2016 release was 83% and for the cleaned corpus and trigrams 85%. Therefore, we use the latter language model. Further, *Cs*=1420 and *Cv*=957 without the additional probabilities of some trigram occurring after a specific trigram vs. *Cs*=1530 and *Cv*=1014 with them; thus, accuracy increased by 7.7% and relevance by 6.0% thanks to this optimisation. Table 1 displays examples of words fed into the corrector and candidate suggestions made by the algorithms.

*Table 1. Illustrative selection of content fed into the corrector and the candidate suggestions provided by the corrector algorithms.*

| Incorrect word | Intended word | Candidate corrections suggested |
|---|---|---|
| Substitution | | |
| *mobani* | *nobani* | *goba nobani ambe bobane emba lobo embi yobe lobu qobo noba gobe nobe amba* |
| *inkosaxana* | *inkosazana* | *inkosi* |
| Insertions | | |
| *imikhuiba* | *imikhuba* | *imikhuba* |
| *zombnuso* | *zombuso* | *zomuntu zomlando zombuso zomu zome* |
| Deletions | | |
| *isaasa* | *isasasa* | *isasasa isale isala isaho isaka* |
| *momo* | *mgomo* | *noma mlomo simo zimo woma zome Zamo dumo zamo komi yoma zomu Noma* |
| Transpositions | | |
| *abezohtakatha* | *abezothakatha* | *abezothakatha* |
| *babemagnaliswa* | *babemangaliswa* | *babe babo baba* |

## *4.1 Non-word error performance evaluation*

Table 2 lists the aggregate results for the four types of single-character errors. Values varied somewhat when disaggregated by the 500, but they all remained within the same range (data not included). The data for the transpositions are those for the optimised version, because that already yielded a *Cs*=472 and *Cv*=424 cf. a non-optimised *Cs*=452 and *Cv*=215 for the first 500 words, i.e., the relevance (*Cv*) being about twice as good with the optimisation. Therewith, the spellchecker's performance for transposition is the best of the four error types.

*Table 2: Main aggregate results for the four spelling errors, with n=1500 inserted errors for each type; TN = true negative,* Cs *= accuracy, and* Cv*= relevance.*

| | *TN (number)* | *Cs (number)* | *Cv (number)* | *Accuracy (%)* | *Relevance (%)* |
|---|---|---|---|---|---|
| *Substitutions* | 1212 | 1110 | 700 | 91.58 | 57.76 |
| *Insertions* | 1448 | 1369 | 434 | 94.54 | 29.97 |
| *Deletions* | 945 | 903 | 669 | 95.56 | 70.79 |
| *Transpositions* | 1444 | 1426 | 1290 | 98.75 | 89.34 |
| *Total* | 5049 | 4808 | 3093 | 95.23 | 61.26 |

For insertion errors, the spellchecker was able to detect incorrectly spelled words, yet *Cv* was low. This could be caused by the extra trigram created when a character is injected into a word and the error corrector does not exclude this trigram when providing candidate corrections. For deletions, the TN values are low and the FP values high, which indicates that when a character is omitted from a word, the trigram constituents of the incorrect word have a high enough frequency that they are all recognized as correct by the error detector.

The threshold frequency may be too low for deletion errors, or a much larger corpus may be required for the spellchecker.

The system's overall performance with the 6000 correctly spelled words and their induced errors is shown in the confusion matrix in Table 3, and the recall and precision measures computed from them. Overall, the system thus performs adequately at detecting and correcting non-word errors.

*Table 3: Confusion matrix with results over the 6000 correct words and their typo-induced variant and the core performance measures computed from them.*

| | *Recognised as correctly spelled* | *Recognised as incorrectly spelled* |
|---|---|---|
| *Correctly spelled word* | TP = 5341 | FN = 659 |
| *Incorrectly spelled word* | FP = 951 | TN = 5049 |

| ***Lexical recall*** | ***Error recall*** | ***Lexical precision*** | ***Error precision*** |
|---|---|---|---|
| ***89%*** | ***84%*** | ***85%*** | ***88%*** |

## *4.2 Comparison evaluation*

The results for error detection among the evaluated tools are shown in Table 4, with notable data in bold: our detector far outperforms Spellchecker.net's detector. Disaggregated results show that the numbers fluctuate considerably even for two samplings of the same corpus and thus would need many more copy-and-paste (and clicking, closing popup boxes, windows, and ads) with manual counts to regress to the mean reliably. The downside of these results is that one cannot reasonably compare corrector accuracy and relevance and draw fair conclusions from it, and this was therefore not pursued. Earlier published results on spellcheckers do not have working tools [2,14] and thus could not be compared.

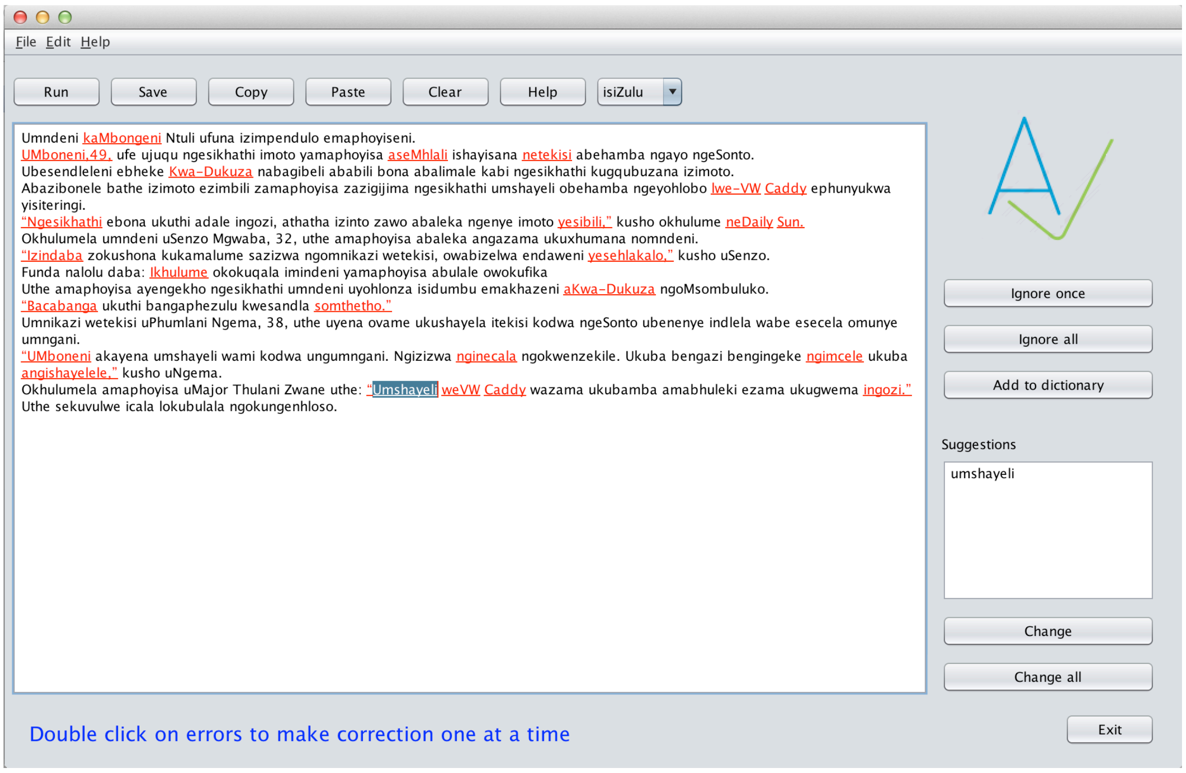
*Table 4. Comparison of three spellcheckers on their error detection capacity on actual pieces of text. N=number of words in the text analysed, SC.net=Spellchecker.net, Pct. LR=percentage lexical recall.*

| *sample* | *N* | *SC.net error* | *Pct. LR* | *2016 release error* | *Pct. LR* | *Error in ours* | *Pct. LR* |
|---|---|---|---|---|---|---|---|
| *UC1* | 172 | 39 | 77.33 | 4 | 97.67 | 8 | 95.35 |
| *Noveli (INC)* | 140 | 21 | 85.00 | 16 | 88.57 | 0 | 100 |
| *NIC 1* | 117 | 18 | 84.62 | 17 | 85.47 | 20 | 82.91 |
| *UC2* | 204 | 27 | 86.76 | 4 | 98.04 | 14 | 93.14 |
| *ilanga news (INC)* | 102 | 20 | 80.39 | 22 | 78.43 | 5 | 95.10 |
| *NIC 2* | 160 | 29 | 81.88 | 28 | 82.50 | 26 | 83.75 |
| sum | 895 | **154** | | 91 | | **73** | |
| avg | | 25.67 | **82.66** | 15.17 | 88.45 | 12.17 | **91.71** |
| median | | 24 | 83.25 | 16.5 | 87.02 | 11 | 94.12 |
| sd | | 7.79 | 3.47 | 9.64 | 8.02 | 9.72 | 6.88 |

## 5. Discussion

The overall result from a statistics-based approach is better than expected, considering that no algorithms for spelling correction are available for isiZulu or any agglutinating Bantu language. It also shows that insightful optimisations, such as the one for transpositions, can have a substantial effect on suggestion adequacy to improve the spelling corrector. Also, the corpus makes a difference, as observed elsewhere as well [11,3,5,18]: just cleaning the data increased the detection accuracy by 2%. It may also greatly help if there were a wordlist of most common words in isiZulu, but this is not known at present. Similarly, knowledge of common typos in isiZulu words and their correction would help the corrector's adequacy, but this is also still to be investigated.

Going a step further toward usable application, we have incorporated the corrector algorithms into a new spellchecker (fn. 8), updating and extending the error detector based on Ndaba et al. [11] (fn. 3). A screenshot is included in Figure 1. This essential step towards practical use of the novel algorithms in business and education brings afore the issue of ordering the candidate corrections, for only a displayable amount of suggestions can be presented to the user. For instance, 1127 of the 1426 words with suggestions for transposition errors had 10 or fewer candidates, which is displayable, but the remaining ones had more than 10 candidates, which somehow have to be ordered and truncated. This will affect the experienced corrector performance by end-users.



Umndeni kaMbongeni Ntuli ufuna izimpendulo emaphoyiseni.
UMboneni,49, ufe ujuqu ngesikhathi imoto yamaphoyisa aseMhlali ishayisana netekisi abehamba ngayo ngeSonto.
Ubesendleleni ebheke Kwa-Dukuza nabagibeli ababili bona abalimale kabi ngesikhathi kugqubuzana izimoto.
Abazibonele bathe izimoto ezimbili zamaphoyisa zazigijima ngesikhathi umshayeli obehamba ngeyohlobo lwe-VW Caddy ephunyukwa yisiteringi.
“Ngesikhathi ebona ukuthi adale ingozi, athatha izinto zawo abaleka ngenye imoto yesibili,” kusho okhulume neDaily Sun.
Okhulumela umndeni uSenzo Mgwaba, 32, uthe amaphoyisa abaleka angazama ukuxhumana nomndeni.
“Izindaba zokushona kukamalume sazizwa ngomnikazi wetekisi, owabizelwa endaweni yesehlakalo,” kusho uSenzo.
Funda nalolu daba: Ikhulume okokuqala imindeni yamaphoyisa abulale owokufika
Uthe amaphoyisa ayengekho ngesikhathi umndeni uyohlonza isidumbu emakhazeni aKwa-Dukuza ngoMsombuluko.
“Bacabanga ukuthi bangaphezulu kwesandla somthetho.”
Umnikazi wetekisi uPhumlani Ngema, 38, uthe uyena ovame ukushayela itekisi kodwa ngeSonto ubenenye indlela wabe esecela omunye umngani.
“UMboneni akayena umshayeli wami kodwa ungumngani. Ngizizwa nginecala ngokwenzekile. Ukuba bengazi bengingeke ngimcele ukuba angishayelele,” kusho uNgema.
Okhulumela amaphoyisa uMajor Thulani Zwane uthe: “Umshayeli weVW Caddy wazama ukubamba amabhuleki ezama ukugwema ingozi.”
Uthe sekuvulwe icala lokubulala ngokungenhloso.

Suggestions
umshayeli

Double click on errors to make correction one at a time

*Figure 1. Screenshot of the new isiZulu spellchecker, which both detects probable single-character non-word errors and provides suggestions for correction.*

Finally, it has to be emphasised that the corpus-driven approach and mostly statistics-based algorithms are exceedingly amenable to bootstrapping a spellchecker for similar agglutinating languages. That is, the new technologies devised for isiZulu are not confined to isiZulu speakers, but with feeding the algorithms text from a related language, one may near-instantly have a fully functional spellchecker for that language. We did this for isiXhosa, which is also integrated in the tool. In one fell swoop, a new spellchecker is available for another 8 million isiXhosa speakers.

In sum, the contributions presented in this paper constitute, to the best of our knowledge, the first set of algorithms for a spelling corrector for isiZulu (or its related languages), which has a fairly good accuracy and can easily be repurposed for related languages. Such technologies are important for redress of the injustices of the past and can provide numerous benefits for learners of said languages, text-based online interactions, as well as serve business and academia in the same way as spellcheckers for other languages already do.

## 6. Conclusions

Novel algorithms have been proposed for isiZulu error correction in spellchecking of non-word errors. Evaluation showed it achieved 89% language recall, 84% error recall, 85% language precision, and 88% error precision for error correction. The overall accuracy is 95% and relevance 61%, which varies much by type of spelling error, with the best results for transpositions (98.8% accuracy and 89.3% relevance). Comparing it to other error correctors was found to be infeasible. The error corrector algorithms have been added to an existing open source isiZulu error detector so as to facilitate immediate use.

It may be possible to add further optimisations, alike done for transpositions, yet others will require research to gain insight into word use and common typos. Bootstrapping the language-independent algorithms with languages similar to isiZulu may be useful as well.

## References

[1] B. Bidyut, A. Chaudhuri. A simple real-word error detection and correction using local word bigram and trigram. Proc. of ROCLING 2013. (Oct 4-5, 2013). ACLCLP, Taiwan, 2013.

[2] S.E. Bosch, R. Eiselen. The effectiveness of morphological rules for an isiZulu spelling checker. S.A.J.Afr.Lang, 2005, 1: 25-36.

[3] K.W. Church, W.A. Gale. Probability scoring for spelling correction. Statistics and Computing, 1, 2 (Dec 1991), 93-103.

[4] N. Farra, N. Tomeh, A. Rozovskaya, N. Habash. Generalized Character-Level Spelling Error Correction. Proc. of ACL 2014. (June 22-27, 2014). ACL, Stroudsburg, Pennsylvania, 2014, 161-167.

[5] A. Hassan, S. Noeman, H. Hassan. Language independent text correction using finite state automata. Proc. of IJCNLP 2008. (Jan 7-12, 2008). AFNLP, Hyderabad, India, 2008.

[6] D. Jurafsky, J.H. Martin,. Spelling Correction and the Noisy Channel; 5, 1.

[7] M.D. Kernighan, K.W. Church, W.A. Gale. A spelling correction program based on a noisy channel model. Proc. of ICCL 1990. ACL, Stroudsburg, Pennsylvania, 1990, 205-210.

[8] L. Khumalo. Advances in Developing corpora in African languages. Kuwala, 2015, 1(2): 21-30.

[9] L. Kukich. Techniques for automatically correcting words in text. ACM Computing Surveys (CSUR), 24, 4 (Dec 1, 1992), 377-439.

[10] R. Mitton. Ordering the suggestions of a spellchecker without using context. Natural Language Engineering, 15, 02 (2009), 173-192.

[11] B. Ndaba, H. Suleman, C.M. Keet, L. Khumalo. The effects of a corpus on isiZulu spellcheckers based on N-grams. In IST-Africa.2016. (May 11-13, 2016). IIMC, Durban, South Africa, 2016, 1-10.

[12] P. Paggio, N.L. Underwood. Validating the TEMAA LE evaluation methodology: a case study on Danish spelling checkers. Natural Language Engineering, 4, 3 (1998), 211-228.

[13] T.A. Pirinen, S. Hardwick. Effect of Language and Error Models on Efficiency of Finite-State Spell-Checking and Correction. In Finite State Methods and Natural Language Processing. (July 23-25, 2012). ACL, Stroudsburg, Pennsylvania, 2012, 1-9.

[14] D.J. Prinsloo, G.-M. de Schryver. Spellcheckers for the South African languages, Part 2: the utilisation of clusters of circumfixes. S.Af.J.Af.Lang. 2004, 1: 83-94.

[15] S. Sharma, S. Gupta. A Correction Model for Real-word Errors. Procedia Computer Science, 70, Supplement C (2015), 99-106.

[16] S. Spiegler, A. van der Spuy, P. Flach. Ukwabelana - An Open-source Morphological Zulu Corpus. Proc. of COLING 2010, 1020–1028

[17] Starlander, M. and Popescu-Belis, A. Corpus-based Evaluation of a French Spelling and Grammar Checker. In Proc. of LREC 2002. May 29-31, 2002.

[18] K. Toutanova, R.C. Moore. Pronunciation modeling for improved spelling correction. Proc. of ACL 2002. ACL, Stroudsburg, Pennsylvania, 2002, 144-151.

[19] G.B. van Huyssteen, E.R. Eiselen, M.J. Puttkammer. Re-evaluating evaluation metrics for spelling checker evaluations. Proc. of COLING 2014. ACL, Stroudsburg, Pennsylvania, 2004, 91-99.

[20] Van Zaanen, M. and Van Huyssteen, G. Improving a spelling checker for Afrikaans. Language and Computers, 47, 1 (2003), 143-156.