Journal of the Digital Humanities Association of Southern Africa, Vol. 5, No.1

Algorithm for assisting grammarians when extracting phonological conditioning rules for Nguni languages

Mahlaza, Zola
Department of Computer Science, University of Cape Town
zmahlaza@cs.uct.ac.za
Khumalo, Langa
SADiLaR, North-West University
Langa.Khumalo@nwu.ac.za

Abstract

Text generation models, the core technology that underpins chatbots such as ChatGPT [1], that are created to support morphologically complex African languages require the modelling of sub-word processes such as phonological conditioning. Since we rely on explicit phonological conditioning rules that are manually identified by grammarians to determine the extent to which such models are able to perform for such languages, there is a need to assist grammarians via computational solutions to increase their coverage of known rules. At present, there are no existing algorithms to extract the rules for such processes and therefore enable the creation of building better text generation models. We present a new algorithm for extracting phonological conditioning rules for Nguni languages. All the rules extracted by the algorithm are valid when the input word and associated morphemes are judged to be valid. The algorithm has the potential to improve the productivity of grammarians and enable the creation of modern text generation technologies that support and promote under-resourced languages.

Keywords: language technologies, low-resource languages, data extraction, phonological conditioning, natural language generation.

1 Introduction

Real-world deployments of Natural Language Generation (NLG) technologies are becoming prevalent. Existing research has demonstrated the potential of these technologies in wide-ranging applications such as healthcare (e.g., Yang et al. (2022)) and education (e.g., Kasneci et al. (2023)). If such technologies are to be of use to individuals in Africa then they will need to also support a variety of low-resourced languages. In the case of South Africa, since a combined 43% of the population speaks either isiZulu, isiXhosa, SiSwati, or isiNdebele as an L1 (see Lehohla (2011)), then it is reasonable to expand the coverage of NLG systems to those languages.

One of the linguistic phenomena that needs to be captured when generating valid text for aforementioned languages is the process of phonological conditioning. To demonstrate the process: the diminutive form of *intaba* ‘mountain’ is formed by appending the suffix *-ana* and in the process, the phonological rule [2] that converts *-b-* to *-tsh-* is activated; hence, *intaba* + *-ana* becomes *intatshana* ‘small mountain’ instead of *intabana*. Since the languages are under-resourced and grammatically complex, prevalent neural text generation architectures are likely to struggle to capture the linguistic phenomena when forming individual words. This is implied by the performance of neural models created to undo phonological conditioning — canonical segmentation in Nguni languages (Moeng et al. (2021)). Moeng et al. (2021)’s best performing canonical segmentation models have F1 scores around 0.7 [3]. However, analysis of the code [4] shows that the true performance is likely to be much lower since the quantification of the performance does not take the order and completeness of the morphemes into account. For instance, when given the word *ezinjeni* ‘in dogs’ and valid canonical segmentation *ez-i-nja-ini*, the implementation would consider the predicted segmentation *nja-i* valid since both its morphemes are found in the valid segmentation (underlined segments).

If the rules were extensively documented in this low-resourced setting, we could make the database of rules accessible to our text generation models, and they would only have to solve the simpler task of deciding whether to apply each rule when combining sub-word elements (instead of also identifying them). In addition, grammarians would have complete understanding of the processes and how they have changed over time. However, the rules are currently not extensively documented because the process of identifying and studying phonological conditioning rules is time-consuming as it is conducted manually by grammarians. There are also no models or algorithms for extracting these rules from the existing datasets of words and canonical morphemes that have been created by computer scientists and grammarians. This presents a barrier to grammarians that are studying the phenomena as an aim in itself and engineers who are attempting to create NLG models.

In this paper, we present an algorithm for extracting phonological conditioning rules for Nguni languages when provided with words and the canonical morphemes. We used the algorithm to extract 298 (isiZulu), 312 (isiXhosa), 308 (siSwati), and 279 (isiNdebele) unique rules. We evaluated the algorithm by randomly sampling 50 rules for each language and **LK**, a grammarian, determined the validity of the input word and canonical morphemes, determined whether they match, and whether the extracted rule is valid. Evaluation showed that 100% of the rules are valid when the input word and morphemes are valid and match for each language. Grammarians can use the algorithm to automatically extract phonological conditioning rules. Engineers can use the extracted rules as input to their neural models to determine the extent to which explicit knowledge improves them.

The rest of the paper is structured such that Section 2 discusses existing work, Section 3 focuses on how to align words and canonical morphemes, Section 4 focuses on the extraction of phonological conditional rules, Section 5 presents evaluation of the algorithm, Section 6 presents the results, Section 7 discusses, and Section 8 concludes.

2 Related work

There are a number of software packages that can be used to study African languages. For instance, WordSmith [5] can be used to study concordances and word forms. Nonetheless, there are no computational tools, models, or algorithms for extracting phonological conditioning rules. As such, we expand our analysis of existing work to also focus on work that codifies, not just extracts, such rules.

There are only three efforts aimed at directly capturing phonological conditioning rules for Niger-Congo B languages. They all either extract the rules from grammar literature, that is often outdated, or source them from a grammarian. These rules are then captured using a programming language. For instance, Keet & Khumalo (2016) use Python to capture rules for isiZulu, the NguniTextGeneration [6] grammar engine uses Java to capture rules for isiZulu and isiXhosa, and Byamugisha (2019) also uses Java to capture rules for Runyankore. The biggest limitation with the approach taken by these authors is that their rules are motivated by use cases (e.g., ontology verbalisation or weather forecast generation), are sourced from dated grammar literature, they are not exhaustive, and do not cover undocumented rules.

When widening the search for related literature further, we see that there exists a verb parser and generator (Pretorius et al. (2017)), morphological analysers (Pretorius & Bosch (2009), Pretorius et al. (2009)), morphological segmenters (e.g., Mzamo et al. (2019), Moeng et al. (2021)), language models (e.g., Myoya et al. (2023)), and a Grammatical Framework (GF) grammar

[7] that also encode morphological conditioning rules, even though that is not their main goal. Specifically, the morphological analysers and canonical segmenters take a word as input and produce canonical morphemes, hence they need to capture phonological conditioning rules, even if that is in an implicit manner, to be able to reverse them. For instance, Moeng et al. (2021)'s models can only generate the canonical morphemes *nga-i-zin-konzo* when given *ngezinkonzo* 'by the services' if they model the reversal of the phonological conditioning rule a + i → e. These resources also cannot uncover new rules and additional limitations to this type of work are as follows:

- The morphological analysers rely on finite state machines that are based on limited and hand-coded rules;
- The morphological segmenters and language models do not model phonological conditioning directly but as part of a larger task, hence they have no identifiable module that can be used for phonological conditioning exclusively; and
- Pretorius et al. (2017)'s GF rules encode phonological conditioning directly but they are limited to Setswana verbs only. The isiZulu resource grammar covers phonological conditioning rules, as part of general morphosyntax, but it is difficult to isolate them as they are not orthogonal.

All in all, there is still a need to create algorithm(s) that can extract phonological conditioning rules from existing canonical segmentation datasets.

3 Aligning words and morphemes

Let Σ denote the alphabet of a language, Σ_s denote the alphabet that includes a special character for separating morphemes (i.e., **-**), and Σ_{sm} denote an extension of Σ_s that includes a symbol to denote a gap in an alignment (i.e., **?**).

Given the alphabet, we can define a word as follows:

Definition 1 (Word) *A word is the row vector $a_{ij} \in \Sigma^{1\times n}$ of length n.*

For instance, the isiZulu term *ezisetshenziswa* 'that are used' is represented using the word [e z i s e t s h e n z i s w a] of length 15. A word can be split into parts to form a canonical morpheme sequence.

Definition 2 (Morpheme sequence) *We define a canonical morpheme sequence of a word $\overrightarrow{w}$ with length n as the row vector $a_{ij} \in \Sigma_s^{1\times m}$ of length m where $m \geq n$.*

For instance, the canonical morpheme sequence of the isiZulu term *ezisetshenziswa* 'that are used' is the vector [e z i **-** s e **-** b e n z **-** i s **-** w **-** a] of length 18. This sequence is not formed in an arbitrary manner. Instead, it is formed by separating a word's linguistic morphemes using the special symbol **-**.

A word and morpheme sequence can be aligned, based on their matching elements, to identify a divergence of characters. The existence of a divergence suggests that characters were introduced by a phonological conditioning rule.

Definition 3 (Word/morpheme alignment) *An alignment of a word $\overrightarrow{w}$ of length n and its canonical morpheme sequence $\overrightarrow{s}$ of length m is the matrix $a_{ij} \in \Sigma_{sm}^{2\times l}$ where $l \geq max\{m, n\}$.*

For instance, a possible alignment of the isiZulu word [e z i s e t s h e n z i s w a] and canonical morpheme sequence [e z i **-** s e **-** b e n z **-** i s **-** w **-** a] is the following matrix: $\begin{bmatrix} e & z & i & \textbf{?} & s & e & \textbf{?} & t & s & h & \textbf{?} & e & n & z & ? & i & s & \textbf{?} & w & \textbf{?} & a \\ e & z & i & \textbf{-} & s & e & \textbf{-} & ? & \textbf{?} & \textbf{?} & b & e & n & z & \textbf{-} & i & s & \textbf{-} & w & \textbf{-} & a \end{bmatrix}$ The above alignment is one of many possible options. We generate all of them by creating an alignment tree via Algorithm 1.

We demonstrate the algorithm's tree construction process using the word [u b u n t u] and canonical segmentation [u b u **-** n t u], visualised in Figure 1. For the input pair, the algorithm con-

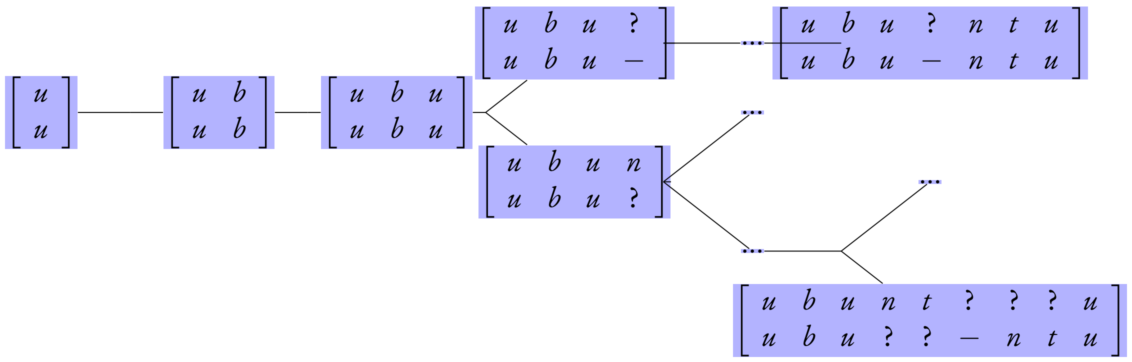

Figure 1: Partial alignment tree where each node captures the strings that have been aligned at each point. The leaf nodes capture all the possible candidate alignments. Ellipsis are used to illustrate abbreviated sub-trees and nodes.

Algorithm 1 Algorithm for building an alignment tree for a sequence of letters

```
 1: procedure COMPLETETREE($left_w$, $left_m$, $curr\_node$)
 2:     $w_i = peekHead(left_w)$
 3:     $m_i = peekHead(left_m)$
 4:     if $w_i == m_i$ then
 5:         $n = Node(curr\_node, w_i, m_i)$
 6:         $completeTree(removeHead(left_w),$
                        $removeHead(left_m), n))$
 7:     else
 8:         $n = Node(curr\_node, w_i, ?)$
 9:         $m = Node(curr\_node, ?, m_i)$
10:         $score^0 = getScore(n)$
11:         $score^1 = getScore(m)$
12:         if $score^0 > score^1$ then
13:             $completeTree(removeHead(left_w), left_m, n)$
14:         else if $score^0 < score^1$ then
15:             $completeTree(left_w, removeHead(left_m), m)$
16:         else
17:             $completeTree(removeHead(left_w), left_m, n)$
18:             $completeTree(left_w, removeHead(left_m), m)$
19:         end if
20:     end if
21: end procedure
```

structs a binary tree using an in-order traversal. The characters of the input are processed one at a time, from left to right (lines 2-3), and if the leading characters of the input are the same (line 4), it constructs a node whose parent is the current node (line 5) and the characters from the word and morpheme sequence are added to the top and bottom of the node's alignment matrix (lines 5).

The process continues with the alignment of the remaining characters (lines 6). When the characters dot not match (lines 7-20), we construct two alternative paths of alignments where in one path we align the word's letter a gap (i.e., **?**) (line 8) and in another we align the morpheme sequence's letter with a gap (i.e., **?**) (line 9). For instance, after aligning the first three letters (highlighted blue) in the case of the word [u b u **n t u**] and morpheme sequence [u b u **- n t u**], third node in Figure 1 (left-to-right), the leading character in the remaining characters [8] of the word and morpheme sequence do not match (i.e., they are *-n-* and **-** respectively). We then create two sub-tree paths. In one sub-tree, we align the *-n-* from the word with a gap (i.e., **?**). For the second sub-tree, we align the morpheme separator letter **-** with a gap (i.e., **?**). This process continues until all letters are aligned. We also make use of a simple heuristic: we do not expand a path if there is a direct alternative path with more matching aligned characters — calculated via *getScore* (lines 10, 11, 12, and 14). The leaf nodes of the constructed tree will contain alignment matrices — two examples are illustrated via the right most nodes in Figure 1.

Once we have extracted the alignment matrices, we then split each matrix using the borders of the morphemes as phonological changes happen in and around those borders.

Definition 4 (Aligned morpheme spans) *The morpheme spans for a given alignment* $\mathbf{A}$*, is the ordered set, denoted* B*, of matrices obtained by splitting the alignment matrix at the points where there are morpheme separators. The set satisfies the following conditions:*

1. $B = \{m_1, m_2, \ldots, m_\alpha\}$
2. $\mathbf{A} = [m_1 \overrightarrow{s}\ m_2 \overrightarrow{s} \ldots \overrightarrow{s}\ m_\alpha]$ *where* $\overrightarrow{s} = \begin{bmatrix} ? \\ - \end{bmatrix}$.

The process of extracting spans is trivial as one only needs to identify positions where symbols **?** and **-** are aligned and then split the matrix. For instance, the set of aligned morpheme spans for the example alignment matrix listed below Definition 3 contains the following elements, in the order in which they are listed (left-to-right, top-to-bottom): $m_1 = \begin{bmatrix} e\ z\ i \\ e\ z\ i \end{bmatrix}$, $m_2 = \begin{bmatrix} s\ e \\ s\ e \end{bmatrix}$, $m_3 = \begin{bmatrix} t\ s\ h\ ?\ e\ n\ z \\ ?\ ?\ ?\ b\ e\ n\ z \end{bmatrix}$, $m_4 = \begin{bmatrix} i\ s \\ i\ s \end{bmatrix}$, $m_5 = \begin{bmatrix} w \\ w \end{bmatrix}$, $m_6 = \begin{bmatrix} a \\ a \end{bmatrix}$

Some of the morpheme spans have gaps in one or both rows (i.e., has the symbol **?**) while others do not. When there are gaps in a span, this indicates that it is likely that there was a change in characters due to phonological conditioning and those spans can be used to extract phonological conditioning rules.

We now turn to focus on the definition of such rules and how to extract them from spans.

4 Retrieving phonological conditioning rules

We use an aligned morpheme span to formally define a phonological conditioning rule. For an alignment matrix, consider the ordered set of all its aligned morpheme spans $\{\mathbf{S_i}, \ldots, \mathbf{S_k}\}$ where each $\mathbf{S} = a_{ij} \in \Sigma_{sm}^{n \times m}$ satisfies the following two conditions:

1. $\exists i, j : a_{ij} = ?$ for some $1 \leq i \leq n, 1 \leq j \leq m$
2. $\exists i, j : a_{i1} \neq ?$ or $a_{1j} \neq ?$ for some $1 \leq i \leq n$, $1 \leq j \leq m$

Stated in another way, we filter out spans such that we are only left with ones that have at least one **?** symbol and one letter on the first or second row on the matrix. For instance, in the case of the spans from our running example, we see that $m_3 = \begin{bmatrix} t\ s\ h\ ?\ e\ n\ z \\ ?\ ?\ ?\ b\ e\ n\ z \end{bmatrix}$ is the only span with at least one **?** symbol and at least one letter. We can then concatenate such spans, while maintaining the order in which they appear in the alignment matrix, to obtain the new matrix $\mathbf{M} = [m_1\ m_2\ \ldots\ m_k]$ and use the matrix to define a phonological conditioning rule.

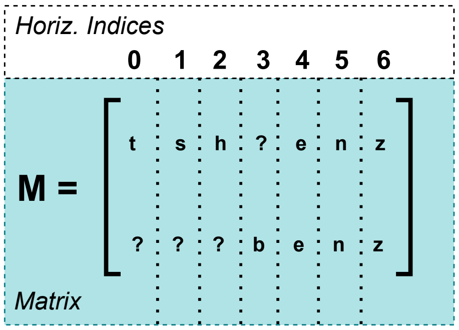

Figure 2: Visualisation of the horizontal (horiz.) indices of the running example's $\mathbf{M}$ *matrix*

Prior to defining these rules, we first introduce a convenient way to index the matrix $\mathbf{M}$. Let $\mathbb{I} = \{\mu(0) \equiv 0, \mu(1), ..., \mu(n)\}$ denote the set with the first horizontal position of the matrix (i.e., $\mu(0)$, its value is set to 0 by definition) and the rest of the horizontal positions of the matrix where the top and bottom values are both letters (i.e., $t_{\mu(i)}, b_{\mu(i)} \in \Sigma$ for $i \in [1 : n]$). We use the symbol μ when denoting these positions since they are relative to the matrix $\mathbf{M}$ and not Definition 3's alignment matrix. For instance, since $\mathbf{M} = m_3$ in our running example, we can deduce — as illustrated in Figure 2 — that $\mathbb{I} = \{0, 4, 5, 6\}$.

Definition 5 (Phonological conditioning rule) *For any consecutive indices* i *and* j *from* $\mathbb{I}$*, the sub-matrix* $\begin{bmatrix} t_{\alpha(i)} & \cdots & t_{\alpha(j)-1} \\ b_{\alpha(i)} & \cdots & b_{\alpha(j)-1} \end{bmatrix}$ *of* $\mathbf{M}$ *is a phonological conditioning rule if the conditions are met:*

1. *For its every horizontal position, the values at the top and bottom cannot both be equal to ?.*

2. *At least two characters in the bottom row of the matrix are letters from the alphabet* Σ

Otherwise, $\begin{bmatrix} t_{\alpha(i)} & \cdots & t_{\alpha(j)-1}, t_{\alpha(j)} \\ b_{\alpha(i)} & \cdots & b_{\alpha(j)-1}, b_{\alpha(j)} \end{bmatrix}$ *is a phonological conditioning if it meets the two conditions.*

We illustrate a phonological conditioning rule using our running example's word [e z i s e t s h e n z i s w a] and associated morpheme sequence [e z i - s e - b e n z - i s - w - a]. As introduced in the previous paragraph, $\mathbb{I} = \{0, 4, 5, 6\}$ for the pair and when considering the first two elements of $\mathbb{I}$ (i.e, $i = 0$ and $j = 4$), then $\begin{bmatrix} t_{\alpha(i)} & \cdots & t_{\alpha(j)-1} \\ b_{\alpha(i)} & \cdots & b_{\alpha(j)-1} \end{bmatrix} = \begin{bmatrix} t_0 & \cdots & t_3 \\ b_0 & \cdots & b_3 \end{bmatrix} = \begin{bmatrix} t & s & h & \textbf{?} \\ \textbf{?} & \textbf{?} & \textbf{?} & b \end{bmatrix}$. This matrix satisfies Definition 5's first rule but not the second rule. On the other hand, $\begin{bmatrix} t_{\alpha(i)} & \cdots & t_{\alpha(j)-1} & t_{\alpha(j)} \\ b_{\alpha(i)} & \cdots & b_{\alpha(j)-1} & b_{\alpha(j)} \end{bmatrix} = \begin{bmatrix} t_0 & \cdots & t_3 & t_4 \\ b_0 & \cdots & b_3 & t_4 \end{bmatrix} = \begin{bmatrix} t & s & h & \textbf{?} & e \\ \textbf{?} & \textbf{?} & \textbf{?} & b & e \end{bmatrix}$ and the sub-matrix satisfies both conditions; hence it is a phonological conditioning rule. This sub-matrix captures a special case of the process called palatalization that is responsible for the transformation of **/b/** to **/tsh/** (i.e., **b** + **e** $\rightarrow$ **tshe** in this case). Using the above definition, we developed Algorithm 2 for extracting the phonological conditioning rules.

The algorithm expects a word and canonical morpheme sequence (line 1) and retrieves the alignments via *getAlignments* (line 3) — a method defined in Algorithm 3. The algorithm retrieves all the leaf nodes, from the alignment tree constructed via Algorithm 1, that have the most matching aligned letters. Using Figure 1 to demonstrate, if the two visualised leaf nodes (right most nodes) were the only ones in the tree then the top node would be chosen since it has 6 matching letters while the bottom node has 4. Algorithm 2 then retrieves the spans for each alignment using the procedure described below Definition 4 and filters spans to select the ones that contain at least one **?** symbol and at least one letter — this is denoted as *getSpans* in the algorithm (line 5). The spans are concatenated to form a matrix **M** (line 6-9). Once matrix **M** is created, the algorithm then iterates over the start of the index and the indices of the matrix where there are letters (lines 10-15) and extracts the phonological rules by checking whether each sub-matrix satisfies Definitions 5's conditions (lines 12-13, 14-15).

Algorithm 2 Algorithm for extracting phonological conditioning rules from a word and its morphological decomposition

```
Require: $w, m$
 1: procedure GETRULES($w, m$)
 2:     $rules \leftarrow []$
 3:     $alignments \leftarrow getAlignments(w, m)$
 4:     for $alignment$ in $alignments$ do
 5:         $spans \leftarrow getSpans(alignment)$
 6:         $M \leftarrow []$
 7:         for $span$ in $spans$ do
 8:             $concatenate(M, span)$
 9:         end for
10:         $I \leftarrow [0] + getIndicesOfLetters(M)$
11:         for consecutive $i, j$ in $I$ do
12:             if M[i,j-1] is phon. cond. rule then
13:                 $rules.add(M[i, j-1])$
14:             else if M[i,j] is phon. cond. rule then
15:                 $rules.add(M[i, j])$
16:             end if
17:         end for
18:     end for
19:     return rules
20: end procedure
```

The algorithm implementation and supplementary data is available from `https://zenodo.org/records/10060974`.

5 Evaluation

We downloaded and combined the SADiLaR-II [9] canonical morpheme segmentation data for Nguni languages and the labelled morpheme segmentation data from the Ukwabelana corpus (Spiegler et al. (2010)). We preprocessed the data in a number of ways. For instance, removed entries where the word is only a punctuation mark (e.g., a full-stop), a foreign word (e.g., the English word "Inductive"), and entries

Algorithm 3 Algorithm for retrieving all possible alignments for a word and canonical morpheme sequence

```
Require: $w, m$                                          # Word and morpheme sequence expected as input
 1: procedure GETALIGNMENTS($w, m$)
 2:     $parent = \emptyset$                              # Create the parent of alignment (binary) tree's root
 3:     $completeTree(w, m, parent)$                      # Construct tree with possible alignments for $w$ and $m$ using Algo. 1
 4:     $max\_nodes = getChildrenNodesWithMostMatchingLetters(root)$   # Get leaf nodes with most aligned matches
 5:     $alignments = []$                                 # Create empty list of alignments
 6:     for $node$ in $max\_nodes$ do
 7:         $t \leftarrow node.getTop()$                  # Letters occupying the top row of the node's alignment matrix
 8:         $b \leftarrow node.getBottom()$               # Letters occupying the top row of the node's alignment matrix
 9:         $alignment = \begin{bmatrix} t \\ b \end{bmatrix}$   # Creating alignment matrix
10:         $alignments.add(alignment)$                   # Collect current alignment extracted from node
11:     end for
12:     return alignments
13: end procedure
```

where the number of each letter is the same between the word and its corresponding canonical segmentation (e.g., ababeke and a-ba-bek-e) [10]. We also generated multiple variations for morpheme segments that use round braces to denote optional segments. For instance, the word *iimpendulo* 'answers' has the canonical segmentation *i-(z)im-pendulo* in the dataset. For this pair, we removed the use of round braces by generating the following two pairs: (*iimpendulo*, *i-zim-pendulo*) and (*iimpendulo*, *i-im-pendulo*). The preprocessed data was fed to the algorithm to extract 298 (isiZulu), 312 (isiXhosa), 308 (siSwati), and 279 (isiNdebele) unique phonological conditioning rules. We sampled and packaged 50 phonological conditioning rules, together with their associated word and morpheme segmentation, for each language. After approval by the ethics committee, the data was evaluated by a grammarian and they were asked the following questions for each combination:

1. *Is the word valid?*
2. *Is the morphological segmentation valid?*
3. *Does the morphological segmentation match the word?*
4. *Is the phonological conditioning rule valid?*

We then calculated the percentage of words, morphemes, and rules that are judged to be (in)valid. In the case where there are invalid rules, we determined whether that is due to the correctness of the word/segmentation or another reason.

6 Results

The results of the expert evaluation are given in Figure 3-6. The decision trees correspond to the questions that the grammarian has to answer. For the first/root cell, we have 50 words for which the grammarian specified *Yes*/*No*/*Uncertain* to the question *Is the word valid?*. The visualisation follows the same process for correctness of the morphological segmentation, whether words and morpheme segmentation match, and correctness of the rule. We trace the top-most path of Figure 4 to demonstrate how to interpret the visualisation, the path shows that the grammarian evaluated 50 words and specified that 2 of them are invalid. For those 2 words, all their corresponding morphological segmentations are invalid and the words do not match the segmentations. Lastly, each rule associated with each segmentation and word pair is judged to be invalid.

For the 50 evaluated rules, 48% (isiZulu), 54% (siSwati), 92% (isiXhosa), and 40% (isiNdebele)

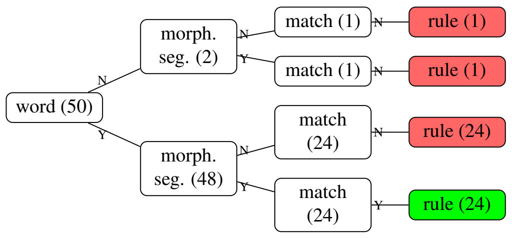

Figure 3: IsiZulu results. Green and red cells denote valid and invalid rules (Y=Yes, N=No, Morph Seg = Morphological segmentation)

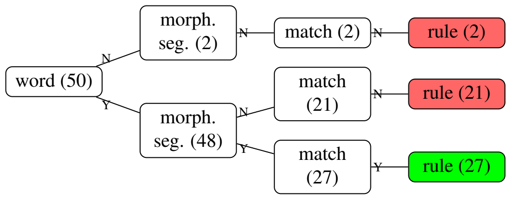

Figure 4: siSwati results. Green and red cells denote valid and invalid rules (Y=Yes, N=No, Morph Seg = Morphological segmentation)

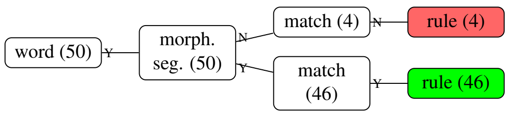

Figure 5: IsiXhosa results. Green and red cells denote valid and invalid rules (Y=Yes, N=No, Morph Seg = Morphological segmentation)

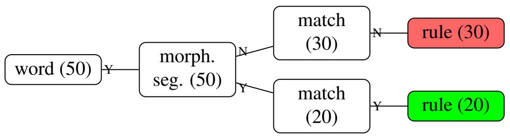

Figure 6: IsiNdebele results. Green and red cells denote valid and invalid rules (Y=Yes, N=No, Morph Seg = Morphological segmentation)

of the rules were judged to be correct. However, 100% of the rules are correct if the word and morphemes are correct and they match – for all languages.

7 Discussion

The results indicate that the algorithm extracts valid phonological conditioning rules based on the validity of the input. In isiZulu, it only fails when the input is incorrect. For instance, when the input is the word and morpheme pair (*soku-1, sa/u/ku/1*). The word *soku-1* is judged to be invalid as it is a shorthand for *sokuqala* 'the first'. In siSwati, it fails when the morpheme segmentation is invalid. For instance, the segmentation of the word *nebesilisa* 'of the male stock' is *na/be/si/lisa* according to the dataset even though the correct segmentation is *ne/be/isi/lisa*. In addition, the morpheme segmentation of *tenkinga* 'problems' is *ta/in/inkinga* in the dataset instead of correct *ta/itiN/nkinga*[11] hence the rule judged to be invalid. In isiXhosa, the rules are only invalid when the morpheme segmentation is invalid. For instance, the morphological segmentation of the word *kokuba* in dataset is given as *kwa/ukuba* instead of the correct segmentation *ko/ku/ba*. In isiNdebele, it also only fails when morphological segmentation is invalid. For instance, for the word *komsebenzi* 'of work', the dataset has the segmentation *kwa/u/mu/sebenzi* instead of *ka/umu/sebenz/i*. Overall, this demonstrates that there is a need to clean up existing morpheme segmentation datasets in order to extract valid phonological conditioning rules.

8 Conclusion

We have created the first algorithm for extracting phonological conditioning rules from data. Evaluation of the unique rules extracted by the algorithm shows that all the extracted rules are correct when the input is valid. We do not foresee any negative societal impacts brought on by the algorithm.

Future work includes the use of dynamic programming to speed up the extraction of the rules, investigating the efficacy of neural models that operate at the sub-word level and have access to the extracted phonological conditioning rules to generate text, creating a word/segmentation ranking module that can be used to filter out invalid input.

Notes

[1] `https://openai.com/blog/chatgpt`. Accessed: 26 July 2023

[2] The process of converting the *-b-* to *-tsh-* in this context is called palatalisation.

[3] The range of the score is 0-1 where the highest possible score is 1.

[4] `https://github.com/DarkPr0digy/MORPH_SEGMENT`. Accessed: 24 July 2023

[5] `https://lexically.net/wordsmith/`. Access date: 26 July 2023

[6] `https://github.com/AdeebNqo/NguniTextGeneration`. Access date: 26 July 2023

[7] `https://github.com/GrammaticalFramework/gf-rgl/tree/master/src/zulu`

[8] The letters that have already been processed are blue and the unprocessed are black and bold.

[9] `https://repo.sadilar.org/handle/20.500.12185/546`. Access date: 26 July 2023

[10] This is done to remove words that are highly unlikely to have no phonological conditioning

[11] Here, the *N* denotes the nasal which can take the form *-n-* or *-m-*.

References

Byamugisha, J. (2019), Ontology verbalization in agglutinating Bantu languages: a study of Runyankore and its generalizability, PhD thesis, Department of Computer Science, University of Cape Town, South Africa.

Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdel, C., Pfeffer, J., Poquet, O., Sailer, M., Schmidt, A., Seidel, T., Stadler, M., Weller, J., Kuhn, J. & Kasneci, G. (2023), 'Chatgpt for good? on opportunities and challenges of large language models for education', *Learning and Individual Differences* **103**, 102274.

Keet, C. M. & Khumalo, L. (2016), On the verbalization patterns of part-whole relations in isiZulu, *in* A. Isard, V. Rieser & D. Gkatzia, eds, 'INLG 2016 - Proceedings of the Ninth International Natural Language Generation Conference, September 5-8, 2016, Edinburgh, UK', The Association for Computer Linguistics, pp. 174–183.

Lehohla, P. (2011), Census 2011: Census in brief, Technical Report 03-01-41, Statistics South Africa. Accessed: 22-June-2022.

Moeng, T., Reay, S., Daniels, A. & Buys, J. (2021), Canonical and surface morphological segmentation for Nguni languages, *in* E. Jembere, A. J. Gerber, S. Viriri & A. W. Pillay, eds, 'Artificial Intelligence Research - Second Southern African Conference, SACAIR 2021, Durban, South Africa, December 6-10, 2021, Proceedings', Vol. 1551 of *Communications in Computer and Information Science*, Springer, pp. 125–139.

Myoya, R. L., Banda, F., Marivate, V. & Modupe, A. (2023), Fine-tuning multilingual pretrained African language models, *in* '4th Workshop on African Natural Language Processing'.

Mzamo, L., Helberg, A. & Bosch, S. (2019), Towards an unsupervised morphological segmenter for isiXhosa, *in* '2019 Southern African Universities Power Engineering Conference/Robotics and Mechatronics/Pattern Recognition Association of South Africa (SAUPEC/RobMech/PRASA)', IEEE, pp. 166–170.

Pretorius, L. & Bosch, S. E. (2009), Finite state morphology of the Nguni language cluster: Modelling and implementation issues, *in* A. Yli-Jyrä, A. Kornai, J. Sakarovitch & B. W. Watson, eds, 'Finite-State Methods and Natural Language Processing, 8th International Workshop, FSMNLP 2009, Pretoria, South Africa, July 21-24, 2009, Revised Selected Papers', Vol. 6062 of *Lecture Notes in Computer Science*, Springer, pp. 123–130.

Pretorius, L., Marais, L. & Berg, A. (2017), 'A GF miniature resource grammar for Tswana: modelling the proper verb', *Lang. Resour. Evaluation* **51**(1), 159–189.

Pretorius, L., Viljoen, B., Pretorius, R. & Berg, A. (2009), A finite state approach to Setswana verb morphology, *in* A. Yli-Jyrä, A. Kornai, J. Sakarovitch & B. W. Watson, eds, 'Finite-State Methods and Natural Language Processing, 8th International Workshop, FSMNLP 2009, Pretoria, South Africa, July 21-24, 2009, Revised Selected Papers', Vol. 6062 of *Lecture Notes in Computer Science*, Springer, pp. 131–138.

Spiegler, S., van der Spuy, A. & Flach, P. A. (2010), Ukwabelana - an open-source morphological Zulu corpus, *in* C. Huang & D. Jurafsky, eds, 'COLING 2010, 23rd International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2010, Beijing, China', Tsinghua University Press, pp. 1020–1028.

Yang, X., Chen, A., PourNejatian, N., Shin, H. C., Smith, K. E., Parisien, C., Compas, C., Martin, C., Costa, A. B., Flores, M. G., Zhang, Y., Magoc, T., Harle, C. A., Lipori, G., Mitchell, D. A., Hogan, W. R., Shenkman, E. A., Bian, J. & Wu, Y. (2022), 'A large language model for electronic health records', *npj Digital Medicine* **5**(1), 194.
URL: *https://doi.org/10.1038/s41746-022-00742-2*