Computational Parsing of Hul’q’umi’num’ Reduplication Patterns:

A Tool for Teaching Reduplication Through Computer Games

Chloë Farr

Department of Linguistics, University of Victoria

LING 590: Hul’q’umi’num’ Morphological Patterns

Dr. Suzanne Urbanczyk

April 26, 2022

Hul’q’umi’num’ is a Central Salish language that has morphological processes unlike those found in English. This paper focuses on reduplication, which is one of the non-concatenative morphological processes found in Hul’q’umi’num’. An example of reduplication marking plural as found in the Hukari & Peter (1995) dictionary can be seen in (1)[1]:

(1)	*Lexeme*	*Parse*	*Gloss*	*HP ID*
	mun’u	√mun’u	offspring: son, daughter	2154
	me’mun’u	√mun’u=PL	offspring: sons, daughters	2158

Reduplication takes on many different forms in Hul’q’umi’num’, and is unfamiliar to learners whose first language is English. On the surface, there also appears to be a lack of predictability in these forms and how they match to meaning, making it challenging to teach them to learners in a comprehensive manner.

The question I seek to provide an answer for is: How can we teach learners about all the different forms of reduplication in Hul’q’umi’num’? In this paper, I will present the initial version of a computational parser of Hul’q’umi’num’ texts that can be turned into a game to teach adult learners reduplication. I will start by introducing the data and give some examples of reduplication in Hul’q’umi’num’. Next, I will describe the problem and my first attempt at the solution, a parser, with illustrations. Finally, I will conclude by discussing games and the limitations of the parser.

Data

Reduplication is the morphophonological process that generates meaning of plurality in Hul’q’umi’num’ as in (1) and (2), and pluractionality as in (3). The data below illustrates a non-exhaustive list of reduplication patterns.

[1] All data in this paper have been retrieved from the newest edition of the Hukari and Peter Cowichan Dictionary, originally published in 1995, which is found at the following link: https://sqwal.hwulmuhwqun.ca/resources/. They are presented in practical orthography used by the Hul’q’umi’num’ Language Academy and Hul’q’umi’num’ Language and Culture Society. They each list an ID, labelled “HP ID = #”. The number corresponds to the lexeme ID as listed in the dictionary .csv file. The following gloss abbreviations are used throughout: PL = plural; DIM = diminutive; DUR= durative; PROG = progressive; RDP = reduplication.

(2)		*Lexeme*	*Parse*	*Gloss*	*HP ID*
		sqwumqwumey’	s=√qwumey’=PL	dogs	3351

(3)		*Lexeme*	*Parse*	*Gloss*	*HP ID*
	a.	puqwpuqwutum	√pqwa=t=m=PL	break (substance) them PL	2631
	b.	paqwpuqwtus	√pqwa=t=DUR=PL	breaking (substance) into pieces (continually)	2625

Reduplication includes that of CV (4a) and CVC segments (4b).

(4)		*Lexeme*	*Parse*	*Gloss*	*HP ID*
	a.	thuthiqut	√thqet=PL	trees, woods, forest	4959
	b.	muq’muq’ut	√muq’=t=PL	swallow repeatedly, swallow PL	2171

Reduplication patterns of these segments can include ablaut in the copy (5a) or base (5b), glottalization

Patterns also include ablaut (5) and resonant aspiration (6).

(5)		*Lexeme*	*Parse*	*Gloss*	*HP ID*
	a.	kwunkwintul	√kwin=tal=PL	fight PL	750
	b.	kwin’kwun’tul’	√kwin=tal=PROG=PL	fighting (pl)	751

(6)		*Lexeme*	*Parse*	*Gloss*	*HP ID*
		hum’um’q’thut	√muq’u=that=PROG=PL	filling selves with food	2183

Patterns co-occur with other patterns like metathesis (7), or l-infixation (8) in diminutive plurals

(7)		*Lexeme*	*Parse*	*Gloss*	*HP ID*
		thuxthuxt	√thxu=t=PL	push lots of things	5024

(8)		*Lexeme*	*Parse*	*Gloss*	*HP ID*
		skwul’a’kwti’	s=√kwaty=DIM=PL	crazy DIM PL	574

The Problem

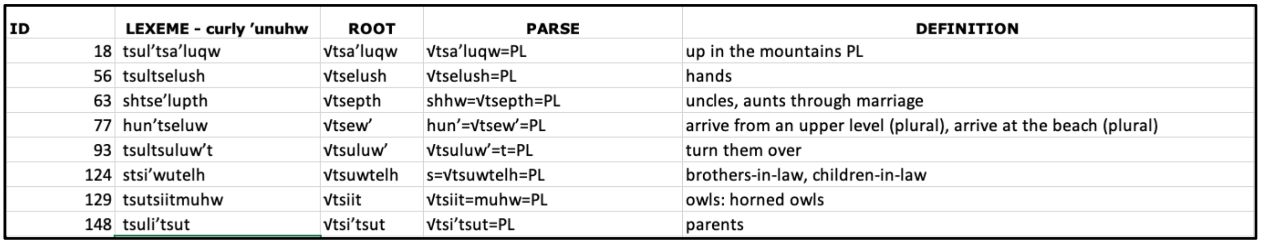
These non-concatenative patterns are opaque and so are potentially unclear to learners (Urbanczyk, 2021). Computer Assisted Language Learning (CALL) can be a useful tool to exemplify grammatical patterns and has been proven to be successful in Indigenous language teaching (Junker et al., 2016; Junker & Torkornoo, 2012; *Teaching First Nations, Métis and Inuit Languages*, 2022). My solution for teaching these opaque patterns is to develop a morphophonological logical parser and computer game.

My Solution

To develop this game, there are two steps: first, identify the different reduplication patterns present in each word and compile them, and second, take the compiled data and read it into the computer game. For the first, I have developed a parsing program to identify morphophonological patterns in a word. Researchers have been actively developing parsers for morphological analysis with Indigenous languages (Assini, 2013; Lonsdale, 2001), and others (Urbanczyk, 2021) are looking for innovative ways to teach these sorts of patterns. The parsing program I present here uses data from the dictionary by Hukari & Peter (1995). An example of the tabular data from the dictionary is seen in Figure 1.

Figure 1

A screenshot of a portion of the Hukari & Peter Parse dictionary, showing a lexeme and its root, parse, and definition.



ID	LEXEME - curly ’unuhw	ROOT	PARSE	DEFINITION
18	tsul’tsa’luqw	√tsa’luqw	√tsa’luqw=PL	up in the mountains PL
56	tsultselush	√tselush	√tselush=PL	hands
63	shtse’lupth	√tsepth	shhw=√tsepth=PL	uncles, aunts through marriage
77	hun’tseluw	√tsew’	hun’=√tsew’=PL	arrive from an upper level (plural), arrive at the beach (plural)
93	tsultsuluw’t	√tsuluw’	√tsuluw’=t=PL	turn them over
124	stsi’wutelh	√tsuwtelh	s=√tsuwtelh=PL	brothers-in-law, children-in-law
129	tsutsiitmuhw	√tsiit	√tsiit=muhw=PL	owls: horned owls
148	tsuli’tsut	√tsi’tsut	√tsi’tsut=PL	parents

Note: Words have been filtered by PL in this example.

The entries in this dictionary include a morphological breakdown of the lexeme, but do not describe the reduplication into a form that shows learners what's going on in terms of morphophonological patterns. The program generates detailed information about word-specific morphophonological processes. The game uses data generated by the parser to illustrate Hul’q’umi’num’ reduplication patterns and to test Hul’q’umi’num’ learners on their understanding the patterns. I will first present the parsing program I have developed, and after, I will present the design of the game.

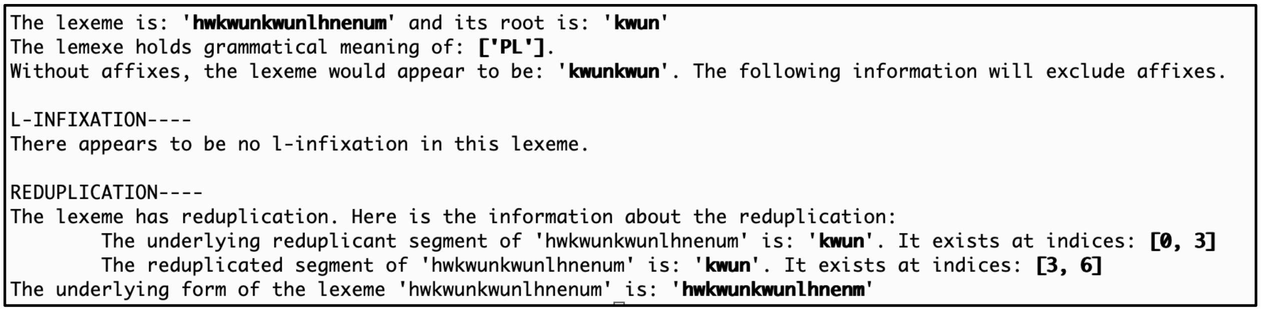
The Parser

The parsing program generates data based on existing theoretical linguistic research on Hul’q’umi’num’ reduplication (Hukari, 1978; Urbanczyk, 2004, 2019, 1998). When given a word and its parse, it returns a summary of the morphophonological patterns the root has undergone to reach the surface lexeme, along with an underlying form of the lexeme. As an example, take ‘hwkwunkwunlhenum’ (9): it has CVC- reduplication. My program takes the input and returns a summary as shown in Figure 2.

	Lexeme	*Parse*	*Gloss*	*HP ID*
(9)	hwkwunkwunlhnenum	hw=√kwun=lhnen=m=PL	hunter, good provider	695

Figure 2

Screenshot of the user-friendly output of information generated by the parser for the word ‘hwkunkwunlhnenum’.



```
The lexeme is: 'hwkwunkwunlhnenum' and its root is: 'kwun'
The lemexe holds grammatical meaning of: ['PL'].
Without affixes, the lexeme would appear to be: 'kwunkwun'. The following information will exclude affixes.

L-INFIXATION----
There appears to be no l-infixation in this lexeme.

REDUPLICATION----
The lexeme has reduplication. Here is the information about the reduplication:
        The underlying reduplicant segment of 'hwkwunkwunlhnenum' is: 'kwun'. It exists at indices: [0, 3]
        The reduplicated segment of 'hwkwunkwunlhnenum' is: 'kwun'. It exists at indices: [3, 6]
The underlying form of the lexeme 'hwkwunkwunlhnenum' is: 'hwkwunkwunlhnenm'
```

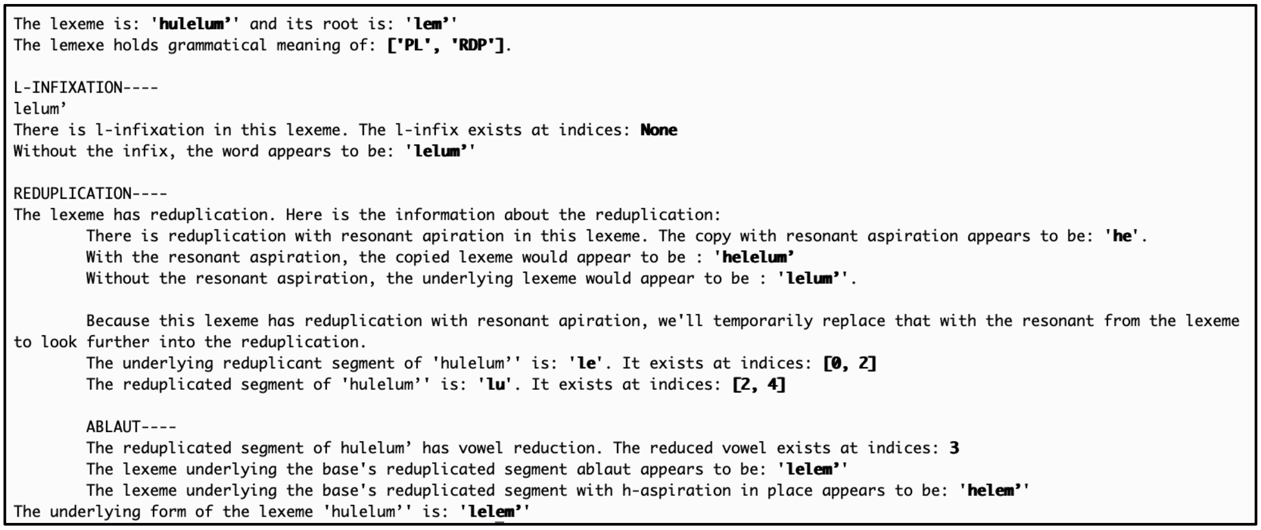
An example with multiple concurrent patterns would be ‘hulelum’’ (10) whose patterns include CV reduplication with reduction of the copy vowel, resonant aspiration on the copy onset consonant, and l-infixation.[2] The output is shown in Figure 3.

	Lexeme	*Parse*	*Gloss*	*HP ID*
(10)	hulelum’	√lem’=RDP=PL	houses	1141

[2] It is assumed that the copy vowel ablauts from /e/ to /i/ based on the ablaut seen in *tsi’tsetl’um’* from the root √*tstl’um* (DIM, PL) which doesn’t contain l-infixation.

Figure 3

Screenshot of the user-friendly output of information generated by the parser for the word 'hulelum''.



```
The lexeme is: 'hulelum'' and its root is: 'lem''
The lemexe holds grammatical meaning of: ['PL', 'RDP'].

L-INFIXATION----
lelum'
There is l-infixation in this lexeme. The l-infix exists at indices: None
Without the infix, the word appears to be: 'lelum''

REDUPLICATION----
The lexeme has reduplication. Here is the information about the reduplication:
        There is reduplication with resonant apiration in this lexeme. The copy with resonant aspiration appears to be: 'he'.
        With the resonant aspiration, the copied lexeme would appear to be : 'helelum'
        Without the resonant aspiration, the underlying lexeme would appear to be : 'lelum''.

        Because this lexeme has reduplication with resonant apiration, we'll temporarily replace that with the resonant from the lexeme
to look further into the reduplication.
        The underlying reduplicant segment of 'hulelum'' is: 'le'. It exists at indices: [0, 2]
        The reduplicated segment of 'hulelum'' is: 'lu'. It exists at indices: [2, 4]

        ABLAUT----
        The reduplicated segment of hulelum' has vowel reduction. The reduced vowel exists at indices: 3
        The lexeme underlying the base's reduplicated segment ablaut appears to be: 'lelem''
        The lexeme underlying the base's reduplicated segment with h-aspiration in place appears to be: 'helem''
The underlying form of the lexeme 'hulelum'' is: 'lelem''
```

The procedure of the program is quite extensive, and a walkthrough and description of each step in a basic case is given in Appendix A.

With this information now generated, it can be shown where any pattern exists in a word, i.e., which graphemes are included and the lexeme's underlying form. In summary, it provides a list of the morphophonological patterns and grammatical categories in a lexeme. This affords learners and teachers the option of sequencing the introduction of patterns and to see how they all interact with one another.

Benefits

This program resolves the complications of non-concatenative morphophonological patterns with the *aid* of an electronic parse dictionary file. As pointed out by Urbanczyk (2011), non-concatenative patterns prevent the ability to store meaning-holding morphemes in a lexicon, eliminating the option of using an electronic dictionary file to parse lexemes by identifying concatenated roots and affixes, in our case for content of a language game. We can use this parser to generate meaning-holding segments on a lexeme-by-lexeme basis.

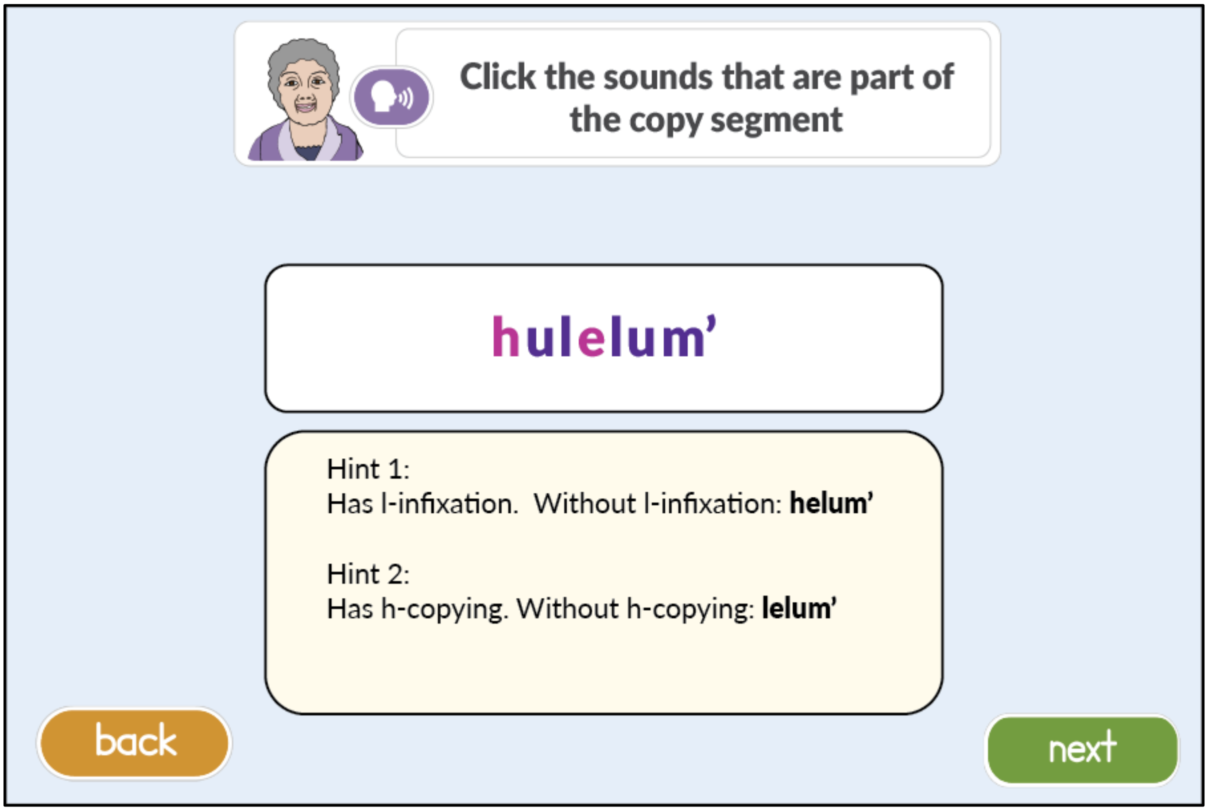
An additional benefit is that, while this program was designed for the purpose of identifying morphophonological patterns in words with reduplication, it has the inherent ability to identify morphophonological patterns in words void of reduplication.

Intended outcome

All of the raw output from the parser can be fed into Unity, a game development software (Unity Technologies, 2021) and turned into a game, like in the mock-up shown in Figure 4, with additional mock-ups shown in Appendix B. In this game, students can identify graphemes that are associated with certain patterns. They can receive hints, and identify how the patterns fit together, such as ablaut in a copy or vowel insertion to accommodate phonological rules.

Figure 4

An example screen for a reduplication game using the morphophonological parser.



Click the sounds that are part of the copy segment

hulelum’

Hint 1:
Has l-infixation. Without l-infixation: **helum’**

Hint 2:
Has h-copying. Without h-copying: **lelum’**

The design of the sections of this game wherein players identify the individual characters involved in each pattern is based on my own intuition. Other sections (shown in Appendix B) where the players identify which patterns exist in a lexeme is based on games that have been developed with the Hul'q'umi'num' Language Academy for teaching grammar (Hul'q'umi'num' Language Academy, 2022).[3]

Discussion

Considerations for the games

Based on personal experience working with teachers and students in the Hul'q'umi'num' Language Academy, run by the Hul'q'umi'num' Language and Culture Society, facilitated by Simon Fraser University, students are engaging with digital activities in a meaningful and effective way. We have been developing these interactive activities (which include videos containing checkpoints with questions, and card-matching memory games) (Farr, 2022a). We have also been developing custom games with responsive features such as feedback based on performance accuracy for learners in the Hul'q'umi'num' Language Academy (aged 18-75 y/o) and the broader Hul'q'umi'num' learning community, such as children (aged 3-6 y/o) in the Language Nest, who play with great enthusiasm (Personal communication, R. Peter, March 18, 2022; Farr, 2022b). At present, these games all focus on lexemes without mention of any non-concatenative patterns.

Considerations for Second Language Teaching

It is important to think about who is playing the game, a question considered in all games I am currently co-developing. Nassaji and Fotos (2011) summarize studies by Jourdenais (1995), Alanen (1995), White (1998) and Simard (2009) and imply that grammar is best learned when learners are provided with explicit enhancements or when combined with explicit instruction, rather than only explicit instruction or neither. The games we are developing are done with consideration of who will be playing them: a student, a group of students, or facilitated by a teacher, and by age. This game will be similarly developed, as the teacher can control the game screen while students interact over Zoom or even in person. The parser is designed not only to generate content that is used by the game, but it is

[3] The referenced website is private and only accessible to students and teachers in the Hul'q'umi'num' Language Academy at present. It will be published after extensive proofreading.

fundamentally designed as a standalone tool that can be embedded into a website and return raw information in a friendly and comprehensible manner, as shown in Figures 2 and 3 above.

For teaching grammar in second languages, Nassaji and Fotos recommend explicit enhancement, which “may be overt form-focused intervention in which the teacher explicitly directs learners’ attention to particular linguistic features through various forms of metalinguistic explanation and rule presentation” (2011, p. 39) without overt metalinguistic explanation. Enhancement may be presented in oral or written form, in which a particular grammatical feature is highlighted. In written form, text may be coloured, bolded and or highlighted. The program identifies not only which processes take place, but details about each, such as which graphemes are associated, and where in the word they exist. This allows for the game to stylize the font of individual graphemes based on the task and the user’s performance in the way that Nassaji and Fotos recommend.

Hugo (2014) references Krashen’s Input Hypothesis, stating “if learners are only exposed to 50 vocabulary words, in a best-case scenario they will only learn those 50 words and perhaps gain some limited morphological or phonological awareness” (p. 97). It would be useful for students to learn the words that they will see in the game prior to playing it. Therefore, it would be recommended that the students learn a large set of words or at least the roots prior to playing the game. Additionally, the parser is beneficial as it draws from a large dictionary containing over 6600 lexemes. By way of the parse, words can be categorized grammatical meaning and morphophonological patterns to aid in bootstrapping.

Considerations for the Parser

Although the parser addresses several the processes, there’s still work to be done, like in (11):

(11)	*Lexeme*	*Parse*	*Gloss*	*HP ID*
a.	pqwat	√pqwa=t	break (substance)	2617
b.	paqwt	√pqwa=t=PROG	breaking (substance) it	2618
c.	puqwpuqwut	√pqwa=t=PL	break (substance) them into pieces	2619
d.	paaqwpuqwut	√pqwa=t=PL=DUR	breaking (substance) into many pieces	2620
e.	pi’paqwt	√pqwa=t=DIM=PROG	breaking (substance) it DIM	2621
f.	pi’pqwat	√pqwa=t=DUR	breaking (substance) it (keep on)	2622
g.	puli’paqwt	√pqwa=t=DIM=PL=PROG	breaking (substance) them DIM	2623
h.	puqwpaqwtus	√pqwa=t=PL=PROG	breaking (substance) them	2624
i.	paqwpuqwtus	√pqwa=t=DUR=PL	breaking (substance) into pieces (continually)	2625

(11) shows many lexemes associated with the root 'pqwa(-t)’. Reduplication and ablaut can be seen in examples (11c-i), each with different ablaut realizations (weakening, lengthening, place shifting) in the copy and or base, along with metathesis (11b, c, f), glottal insertion (11e-g) and l-infixation (11g). The initial program currently looks at each lexeme and its identified parse as an isolated pair to return the patterns that have surfaced from the root and can categorize the lexeme by its collective meaning at the surface, but it doesn’t yet identify individual changes to meaning. However, ablaut of any vowel may have a profound impact on meaning. As seen in the lexemes derived from root, an example of a question that the parser can’t currently answer is:

- How does the reduction of /a/ -> /u/ alter the lexeme’s meaning?

The answer is complex and appears to be: The reduction of /a/ to /u/ *and* metathesis of /qw/ and /a/, followed by reduplication indicates plurality, *however*, if there is additional durative meaning, the copy vowel will strengthen to /a/ or /aa/, and if there is progressive meaning, the base vowel will strengthen to /a/.

Any computational linguistic program or natural language processing tool will be subpar to a trained linguist’s eye as it doesn’t have the language acquisition device or universal grammar that humans do (Chomsky, 2021). However, we can continue to develop this parser to catch up with the research which has been conducted and as research continues. To advance the accuracy with which processes are identified, next steps will incorporate the cross-referencing of multiple lexemes’ [from the same root] processes and meanings to better deduct the meaning of a certain segment, or rather, the specific process it has undergone. Cross-referencing each lexeme, their morphophonological patterns and meanings can provide more insight to the meaning held by individual vowel change in a lexeme on a case-by-case basis and potentially language-wide basis.

Conclusion

Computer games are proving to be beneficial for Indigenous language learners. Non-concatenative patterns can create the need for detailed identification of patterns on a word-by-word basis which can be a time-consuming task. By creating a parser, the labor required for such a task is drastically reduced and allows for more dynamic games to be created with fewer resources.

We should take full advantage of the ability to create programs like this and use them to develop games that support language learners.

Word Count: 2437 Excluding References and Appendices.

References

Assini, A. A. (2013). *Natural language processing and the Mohawk language creating a finite state morphological parser of Mohawk formal nouns* [University of Limerick]. https://ulir.ul.ie/bitstream/handle/10344/3574/assini_2013_natural.pdf?sequence=5

Chomsky, N. (2021, March 26). *Geniune Explanation* [Colloquium Presentation]. Lansdowne Lecture in Linguistics. Online. University of Victoria.

Farr, C. (2022a, March 22). *Computer games for Hul'q'umi'num' language learning.* [Conference Presentation]. Gathering of Hul'q'umi'num' Language Educators. [Online]. Hul'qumi'num' Language and Culture Society.

Farr, C., Parent, M. (2022a, March 25). *'iyus mumun'lh website and computer games.* [Conference Presentation]. Happy Little Ones Language Nest Workshop. [Online]. Hul'qumi'num' Language and Culture Society.

Hugo, R. (2014). Endangered languages, technology and learning: Immediate applications and long-term considerations. In M. Jones (Ed.), *Endangered Languages and New Technologies* (pp. 95–111). Cambridge University Press. https://doi.org/10.1017/CBO9781107279063.009

Hukari, T. E. (1978). Halkomelem nonsegmental morphology. *Proceedings of the 13th International Conference on Salish and Neighbouring Languages.*, 157–209. http://lingpapers.sites.olt.ubc.ca/files/2018/03/1978_Hukari.pdf

Hukari, T. E., & Peter, R. (1995). *The Cowichan dictionary of the Hul'q'umi'num' dialect of the Coast Salish people.* https://sqwal.hwulmuhwqun.ca/resources/

Hul'q'umi'num' Language Academy. (2022). *INLG 300 Grammar*. Hul'q'umi'num' Sqwal.

Junker, M.-O., Mollen, Y., St-Onge, H., & Torkornoo, D. (2016). Integrated web tools for Innu language maintenance. *Papers of the Algonquian Conference; Albany*, *44*, 192.

Junker, M.-O., & Torkornoo, D. (2012). Online language games for endangered languages. *4th International Conference on Education and New Learning Technologies*, 6662–6673. http://www.marieodilejunker.ca/pdf/EDULEARN12-JunkerTorkornoo.pdf

Lonsdale, D. (2001). A two-level implementation for Lushootseed morphology. In L. Bar-el, L. Tamburri Watt, & I. Wilson (Eds.), *International Conference on Salish and Neighbouring Languages* (pp. 203–214). University of British Columbia. https://lingpapers.sites.olt.ubc.ca/files/2018/03/2001_Lonsdale.pdf

Nassaji, H., & Fotos, S. (2011). *Teaching grammar in second language classrooms: Integrating form-focused instruction in communicative context*. Routledge.

Teaching First Nations, Métis and Inuit languages. (2022). University of Alberta. https://guides.library.ualberta.ca/teaching-first-nations-metis-and-inuit-languages/games

Unity Technologies. (2021). *Unity Engine* (2019.4.2f1). Unity Technologies. https://unity.com/

Urbanczyk, S. (2004). Plurality and ablaut in Central Salish. In D. Gerdts & L. Matthewson (Eds.), *Studies in Salish Linguistics in honour of M. Dale Kinkade. UM occasional papers in linguistics*.

Urbanczyk, S. (2011). Evidence from Halkomelem for word-based morphology. *ICSNL*, *XLVI*, 470–489. http://lingpapers.sites.olt.ubc.ca/files/2018/01/2011_Urbanczyk.pdf

Urbanczyk, S. (2019). Prosodic morphology and reduplication. In D. Siddiqi, M. Barrie, C. Gillon, J. D. Haugen, & É. Mathieu (Eds.), *The Routledge handbook of North American languages* (1st ed., pp. 91–113). Routledge. https://doi.org/10.4324/9781315210636-4

Urbanczyk, S. (2021). A note on how weaving and knitting can enhance learning Salish reduplication patterns. *Working Papers of the Linguistics Circle of the University of Victoria*, *31*(1), 148–156. https://journals.uvic.ca/index.php/WPLC/article/view/20190

Urbanczyk, S. (1998). A-templatic reduplication in Halq'emeylem. In K. Shahin, S. Blake, & E. S. Kim (Eds.), *The Proceedings of the Seventeenth West Coast Conference on Formal Linguistics XVII* (pp. 655–669). Stanford, CA: Center Study Language and Information Publications.

Appendix A

The following is a description of the procedure of the parsing program.

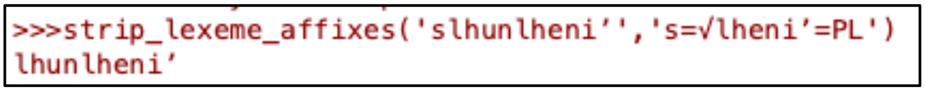
How It Works: The Basics

Preparing the Data.

- The user types a lexeme and its root into the parser.[4]
- The lexeme is stripped of its suffixes and affixes as they are not involved in the reduplication process.[5]

Figure 5

Screenshot of a function call and its output for the function obtaining a lexeme without its affixes.

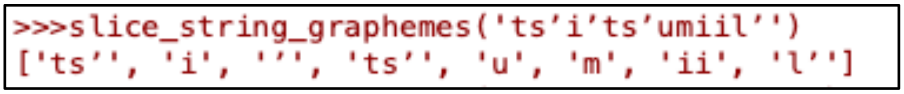


```
>>>strip_lexeme_affixes('slhunlheni’','s=√lheni’=PL')
lhunlheni’
```

- The lexeme is split into graphemes to prevent confusion between sounds such as /l/, /lh/ and /th/.

Figure 6

Screenshot of a function call and its output for the function splitting a lexeme into individual graphemes.



```
>>>slice_string_graphemes('ts’i’ts’umiil’')
['ts’', 'i', '’', 'ts’', 'u', 'm', 'ii', 'l’']
```

Ultimately, we want to know what processes occur, and on which segments, and where those are in the lexeme.

Identifying Patterns.

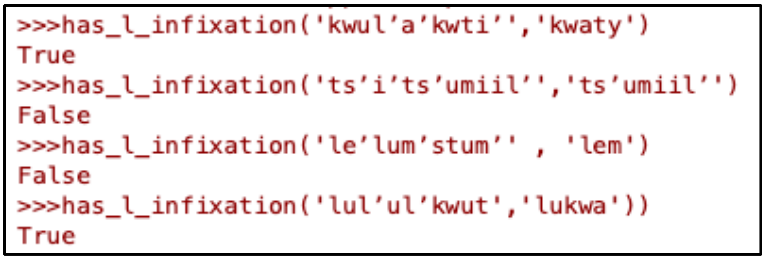
- Identifying l-infixation: the number of /l/s are counted in each the lexeme and the root. If there is one more /l/ in the reduplicant than the root, it returns that l-infixation is present.

Figure 7

Screenshot of a function call and its output for the function identifying the presence of l-infixation.

[4] In the next version, the parser will search the dictionary for the parse so the user doesn’t have to enter it. It currently takes lexemes in Practical Orthography but will be extended to also take APA.

[5] At least, as far as I’ve found. The only impact on a suffix would be ablaut to adhere to a phonological rule.

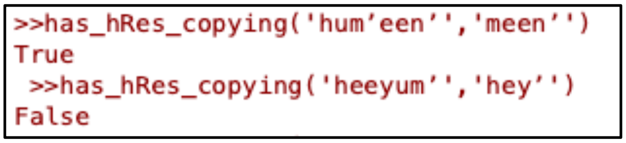


```
>>>has_l_infixation('kwul’a’kwti’','kwaty')
True
>>>has_l_infixation('ts’i’ts’umiil’','ts’umiil’')
False
>>>has_l_infixation('le’lum’stum’' , 'lem')
False
>>>has_l_infixation('lul’ul’kwut','lukwa'))
True
```

- Identifying resonant aspiration: The first graphemes in the stripped lexeme must identify as /h/ or /h’/ and the first graphemes in the root must identify as a non-h(’) resonant.
    - Identifying copying with resonant aspiration: the lexeme is checked for both resonant aspiration and copying.

Figure 8

Screenshot of a function call and its output for the function identifying the presence of resonant aspiration in a copy segment.



```
>>has_hRes_copying('hum’een’','meen’')
True
 >>has_hRes_copying('heeyum’','hey’')
False
```

- To identify reduplication, the graphemes in the stripped lexeme are compared to those in the root, iteratively left to right. Until the graphemes don’t match, they are added to a string (series of characters) that represents the copy aka. ‘copy string’.
    - There is a check for resonant aspiration and l-infixation. If resonant aspiration exists, it is temporarily replaced with the base resonant. If l-infixation exists, the segment is temporarily removed.
    - There is a check for vowel changes, meaning that the vowels don’t have to match, as long as the graphemes are both identified as vowels.
    - There is a check for metathesis.
    - There’s a check for glottal shifting and vowel insertion or deletion, so subsequent graphemes can still be compared.

        - For example, the first vowel in the root 'tsa’luqw' is followed by a glottal stop, but the glottal stop is deleted and the /l/ is now glottalized in the copy segment of the lexeme 'tsul’tsa’luqw'. This creates a disparity in the number of graphemes in the copy and the copied segments, the copy having three graphemes and the copied having four. Without this check, the /l/ would be compared with the unuhw /’/ and wrongly excluded from the copy.
    - Checks that this new ‘copy string’ occurs twice consecutively in the word.
- Identifying ablaut: There are many functions to identify ablaut. There is one function for each of the following: reduction in the copy segment, reduction in the copied segment, strengthening in the copy segment, strengthening in the copied segment, vowel change in the copy segment, vowel change in the copied segment. The functions for vowel change exclude those involving reduction and strengthening, only addressing length and place ablaut. To check for ablaut, the first vowel in the respective segment in the lexeme is individually compared to the first vowel in the root and returns True or False as per the function’s purpose.

Yet to be programed is glottalization and metathesis, therefore some of the functions don’t work on some cases.

Retrieving Segment Information: graphemes and their place.

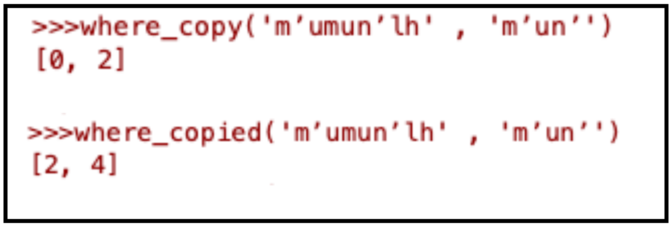
Retrieving the place and graphemes of segments involved in each pattern follows the same processes as with identifying the pattern’s existence. In the functions which identify patterns, the parser only returns True or False for the respective presence or absence of the pattern.

- To retrieve segment place information, the function records the indices while it compares graphemes.[6] In the information displayed to the user, the place information is situated within the original lexeme, accounting for the existence of the affixes in the original lexeme, shown in Figure 9.

[6] Incides are the numerical place values of the grapheme within a word; the first index of a string is 0, and the final index is one beyond that included in the string.

Figure 9

Screenshot of a function call and its output for the function identifying the location of the graphemes included in a copy segment.



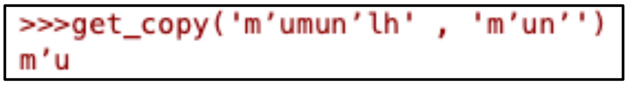
```
>>>where_copy('m’umun’lh' , 'm’un’')
[0, 2]

>>>where_copied('m’umun’lh' , 'm’un’')
[2, 4]
```

- For each to retrieve segments associated with patterns present, the function takes the word, retrieves the indices in which the segment occurs in the words, and returns the graphemes at those indices, shown in Figure 10.

Figure 10

Screenshot of a function call and its output for the function returning the copy segment as a string.



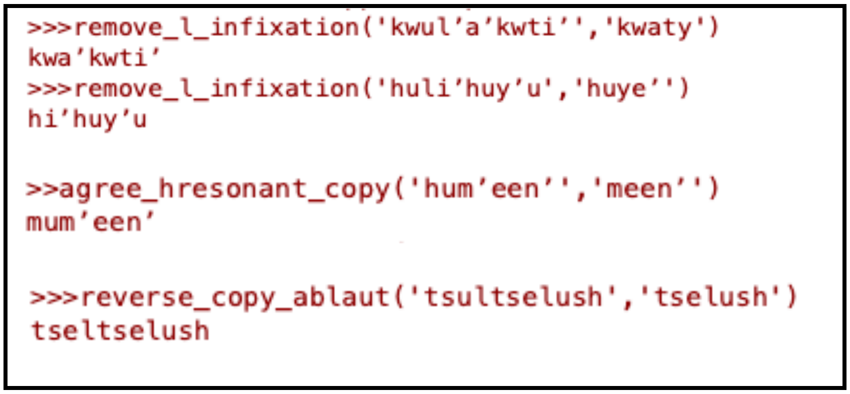
```
>>>get_copy('m’umun’lh' , 'm’un’')
m’u
```

Retrieving Underlying Forms.

For each pattern, there’s a function to remove the associated segment or assimilate it to the root, shown in Figure 11.

Figure 11

Screenshot of a function call and its output for the function returning the lexeme with the l-infix removed.



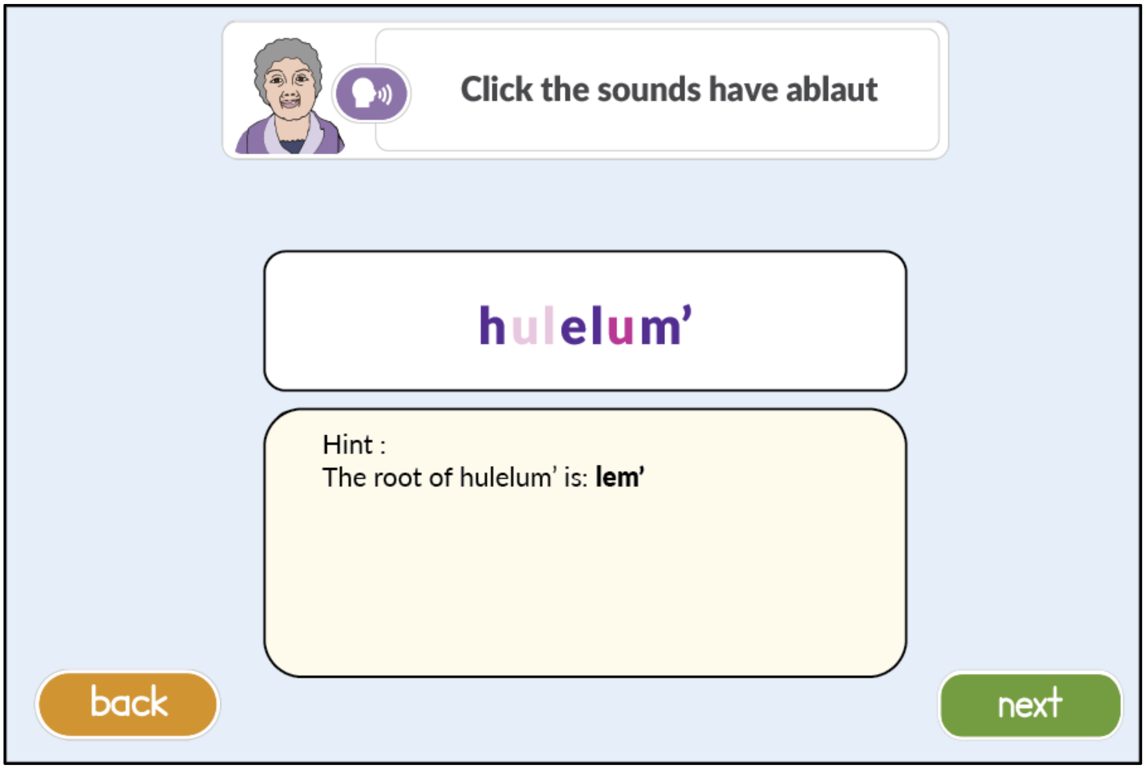
```
>>>remove_l_infixation('kwul’a’kwti’','kwaty')
kwa’kwti’
>>>remove_l_infixation('huli’huy’u','huye’')
hi’huy’u

>>agree_hresonant_copy('hum’een’','meen’')
mum’een’

>>>reverse_copy_ablaut('tsultselush','tselush')
tseltselush
```

Appendix B

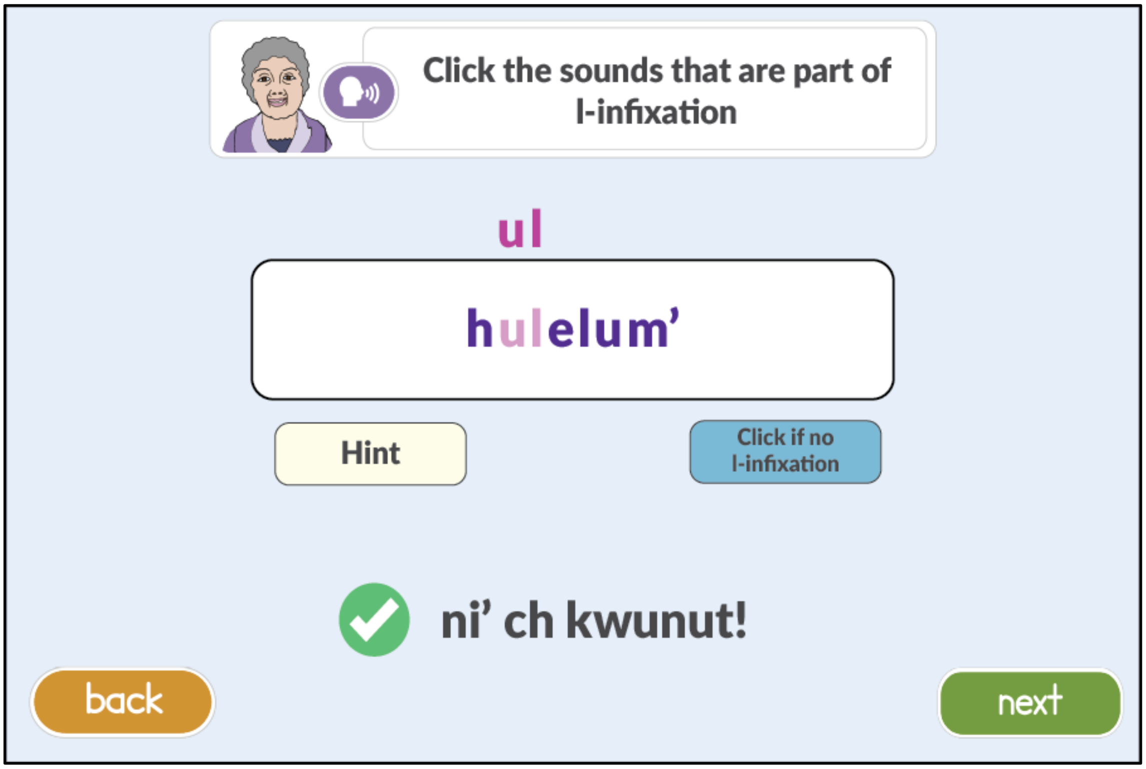
Screenshots of Mock-up Screens for a Game to Teach Reduplication



Click the sounds have ablaut

hulelum’

Hint :
The root of hulelum’ is: **lem’**

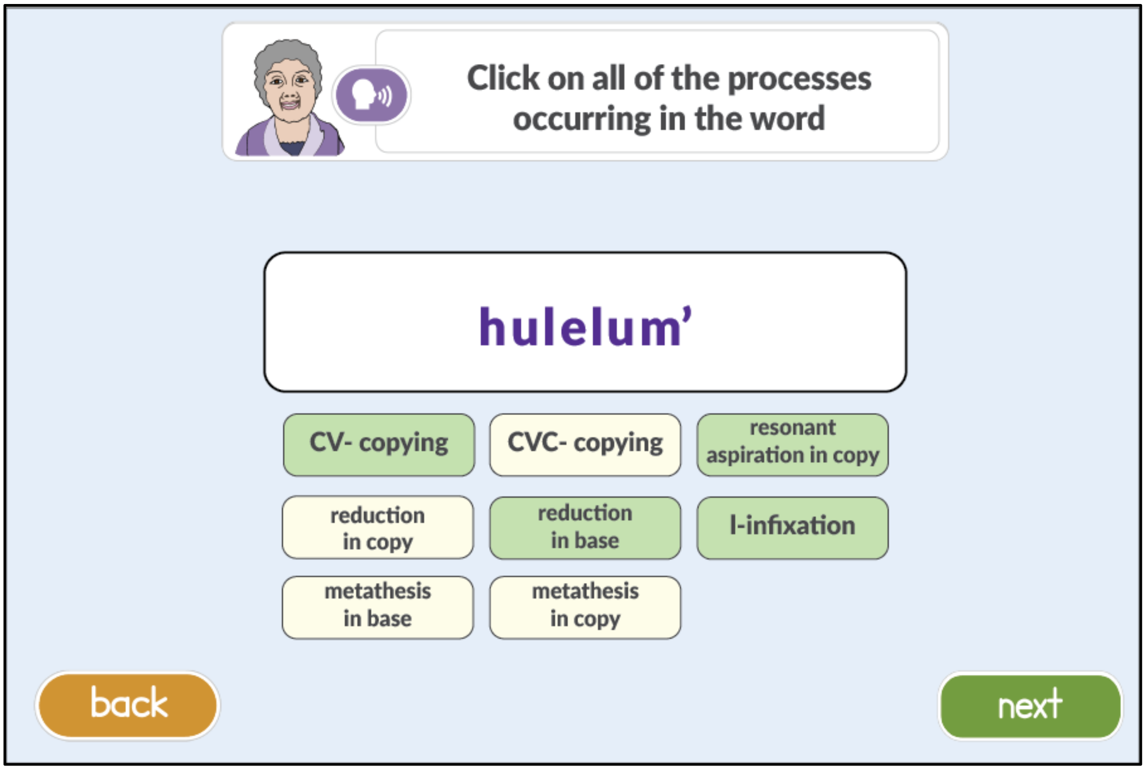


Click the sounds that are part of l-infixation

ul

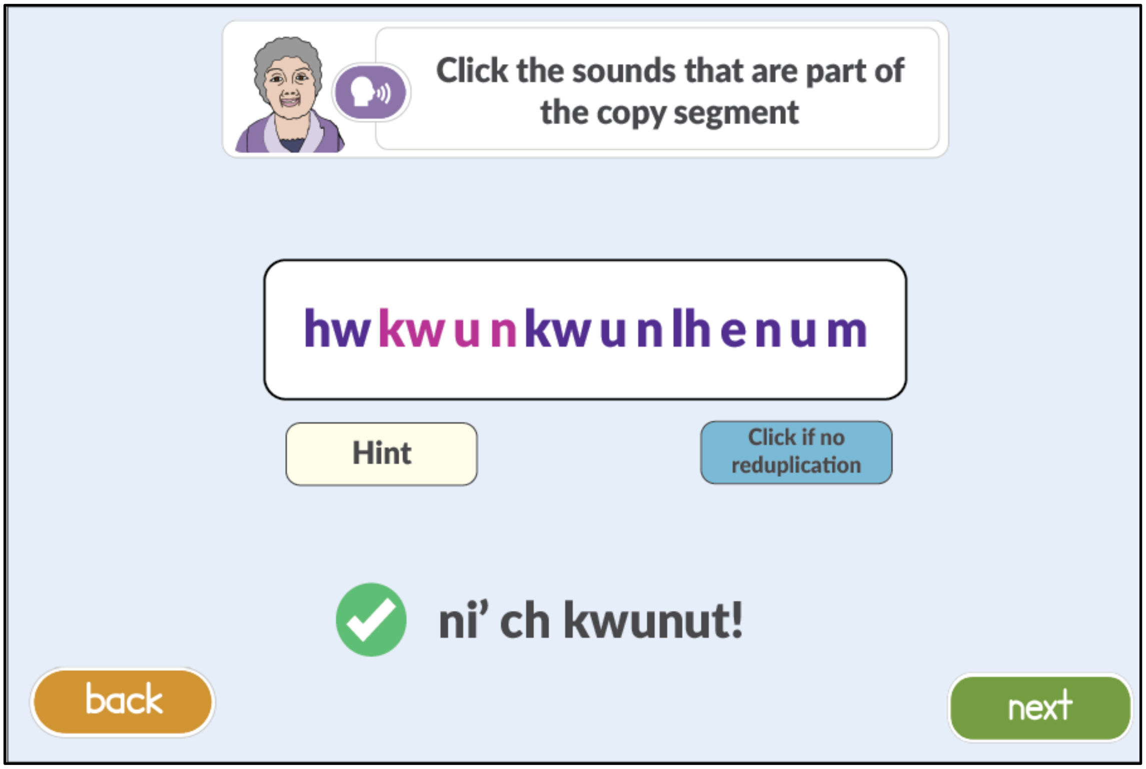
hulelum’

ni’ ch kwunut!



Click on all of the processes occurring in the word

hulelum’



Click the sounds that are part of the copy segment

hw kw u n kw u n lh e n u m

ni’ ch kwunut!