A Minimal Rule Generalization Algorithm for Multi-Part Phonological Rule Learning[1]

Kalina Kostyszyn

Advised by Jane Chandlee

Submitted in partial fulfillment of the requirements for

the degree of Bachelor of Arts in Linguistics

Bryn Mawr College, 2016

[1] I would like to thank Professor Jane Chandlee for her help in advising me during the writing of this thesis, helping me find a topic and guiding me as I connected pieces of what I found and directed my research. I also thank Professor Deepak Kumar for being my second reader and providing valuable feedback in the structure of this thesis. Finally, I thank Sophie Rehrig for being my student reader.

Abstract

In the late 1990's, Adam Albright and Bruce Hayes worked together on an algorithm that learned morphologic patterns from input data, intending to use this as a model for how humans learn morphological and phonemic changes across languages as well. Though the system was effective, it had some shortcomings – the algorithm was not designed to handle some more complicated relationships between morphology and phonology. One such relationship is known as a derived environment, created when the addition of some morpheme generates a phonological pattern deemed unacceptable in that given language. When analyzing a word pair that contained the result of one of these environments, the algorithm reads the resulting change as an original part of the affix, without regard for underlying forms.

Here, I examine the structure of the original algorithm and lay out a method for how future modifications can implement a system to account for theoretical derived environments. This requires splitting the area of change into a substructure that breaks down into two segments: a section that originally is introduced by the affix, and a section that, given a feature-matched phone in the root, likely was transformed after the morpheme was added. To give an example in Polish, for the root word [vag] ('weight'), the result after applying a verb-denoting suffix (here, the suffix [ɨt͡ʃ]) is [vaʒɨt͡ʃ] ('to weigh'). Where the original algorithm would read [ʒɨt͡ʃ] as the entire suffix, the modified version breaks it into the 'original' suffix [ɨt͡ʃ] and the segment [ʒ], the latter of which maps onto [g] in the original word. From this, a rule describing this particular phonological change is generated in addition to a rule describing the affixation.

Table of Contents

1. Introduction

When learning languages from birth, humans undergo a complicated process of repeatedly being exposed to forms of words and trying to replicate them, all in order to perfect their grasp of morphological and phonological rules in their native languages. In fractions of a second, humans can generally be exposed to a new word and determine what feels like a 'correct' way to conjugate or decline it, based on phonological features of the word and how they compare to words we have seen before. To replicate processes like these, phonological and morphological models were created to imitate human cognition and provide greater insight to how, computationally, a human might make these connections regarding their language. The Albright-Hayes Minimal Generalization Algorithm, for example, is given a set of data with some connected word pairs, such as present to past tense of verbs, and determines what speakers are able to learn about the phonology and morphology of the language through tracking transformations between pairs (Albright and Hayes1999).

While they successfully determined morphological and phonological rules for their test languages (English, German, and a constructed 'pseudo-Finnish' were most prominent in their report), Albright and Hayes also noted that, in their trials, there were some languages whose speakers exhibited particular properties about their languages that the algorithm could not yet express. Specifically, they tested some native Polish speakers with affixes that did not previously exist in the language, to see what resulted from the combination of these new affixes and existing words. In fact, the trials found that these speakers were able to do more than change the affix depending on the root word to follow basic phonological rules. These test speakers also enacted secondary changes on the root word depending on this first change, which created previously undocumented phonological rules that nonetheless fell in line with known patterns. Albright and

Hayes determined that speakers were able to create new derived environments, instances where application of one rule when applying an affix created an environment where a second, environment-dependent rule was triggered, rendering the affix insufficient because of a conflict with the root. The speaker then corrects this infraction by applying a second change so that the resulting word is phonologically sound.

Because of limitations imposed by the algorithm Albright and Hayes originally created, they were unable to model these changes; the algorithm as it stood could only determine one change at a time and would thus give a flawed ruleset. However, they hoped to, at some point, follow up on these investigations to see if, by modeling derived environment changes in an algorithm, they could give "psychological reality" to the idea, or demonstrate that this is a learnable process that yields legitimate results. To do this, one must create a system of determining multiple rules from a single entry - if the original algorithm created or modified a single rule based on input, then to accomplish this modified goal, the revised algorithm must determine both a rule for the affix being added and a rule to modify the root based on conflict with the affix. This will prove to be a more intricate process, complicated by difficulty in determining where the root ends after the derived change and where the applied affix begins; however, when coded, this method will give much deeper insight into a language's phonological processes and what happens to a word's phonology when an affix is applied.

2. Minimal Generalization Learner

The study on which I base my modifications is an automated learner for phonological rules based on morphological changes (Albright and Hayes 1999). Given a phonetically-written data set of present tense bare verb and past tense verbs, past tense forms are observed and analyzed with regard to the present-tense root form to develop a basic set of phonological rules. Once the rule set is generated, the algorithm takes in more present tense verbs without the associated past tense and instead generates and outputs a past tense, as determined by the rule set, for each individual input. This algorithm works based on three main principles: mutual bootstrapping of phonology and morphology, minimal generalization, and hypothesis growth plus selection.

2.1 Mutual Bootstrapping

The first concept is the basic idea that one can learn about phonology from morphological changes, and about morphological change from phonological rules, as well as the limitations each imposes on the other. This makes logical sense: if we apply some affix to a word, the resulting combination must be in line with some phonological rules in the language. If the result is identical to the original input plus the suffix, no change was necessary - the underlying form of root plus affix matches the surface form. However, if this is not the case, then some phonological rule was triggered, inflecting the surface form so that phones in the underlying form were subtracted, modified, or even added. For example, in English, we put a verb into past tense by adding the sound [t] or [d] to the end, as in [pikd] However, to put *say* into past tense, rather than simply adding the suffix to create [seɪd] or [seɪt] – instead, we change the vowel entirely to create [sɛd]. Morphologically, this uses the same suffix, but we additionally see that the vowel sequence in the present tense form of the word cannot be followed by [d] or [t] as a past tense

verb. Further, this is tied back to morphology because the word *aid* is pronounced [eɪd], which that the sequence is legal in some morphological contexts – just not as a past tense verb. These inflections help demonstrate an underlying complexity revealed by the interplay of morphology and phonology.

Further, for the algorithm to operate, it creates a number of possible forms of the base word with the specific morphology applied, to be elaborated on later. When generating these possible forms, the least likely forms are removed when they are determined to be ill-formed - that is, they break some phonotactic constraint in the language that means this form does not exist in the language. For example, as established, we make *say* into past tense by adding a suffix and changing the vowel, resulting in [sɛd]. In terms of the transition from present tense to past tense, [seɪd] is considered to be ill-formed, because when [d] is a suffix, the sequence [eɪd] breaks a phonotactic constraint – thus, it is removed. By removing poorly formed possible matches, we are better able to reliably choose the most likely match. Using existing phonology to eliminate impossible morphological matches can, in turn, strengthen the phonological ruleset based on the morphemes that are added.

2.2 Minimal Generalization

The second principle is more complex - when determining rules for a given phonological change, if a set of rules share some common features, these rules are then combined to retain the common features and discard the excess, unshared phones. It is likely, then, that the phonological change at hand is more relevant to these shared features than to features that are present in one rule but not another - if the rule holds without those particular features in place, then they must be irrelevant to the rule itself.

One method of accomplishing this is by simply adding rules together to find common threads between alternations (Albright and Hayes 2002) - in a sense, adding to subtract. This breaks the word being affected into five basic parts: a beginning variable for everything preceding the domain of influence of the change, a series of sets of features for each phone common to the pair of words and occurring before the place of change, the place of change itself, a second series of feature sets that *follow* the place of change, and an ending variable for everything following this domain of influence. To minimally generalize these rules, one must match the site of change for a pair of rules or words and then expand outward to the bounds of each rule until the phones being compared have no matching features. This ensures that the most comparisons are made directly at the site of change, and as the phone comparisons radiate outward, phones are less and less likely to have matching features, until the beginning and ending variables encapsulate a discarded phone.

For example, suppose we have two pairs of words: *miss/missed* and *kick/kicked*. From these pairs, we generate two rules:

(1) null → [t] / [mɪs] _ #

(2) null → [t] / [kɪk] _ #

In both cases, our rule takes an empty space in the word (the ending) and adds the suffix [t] to the end. Since both of these rules are word-final, our resulting rule must also be word-final, so nothing changes after the site of affixation. However, for the phones that precede the affix, we start at the two closest phones - [s] and [k] - and compare them for relevant features. To simplify the process, we will only consider the voicing of consonants; because both [s] and [k] are voiceless consonants, we reduce this phone to the feature [-voice]. The next phones, [ɪ] and [ɪ] are identical, so this phone is retained for the final rule. And for the final phones, [m] is voiced

while [k] is not. Because these two phones differ on this crucial feature, it is replaced entirely by the variable X, denoting that any phone can take this place. Since this is the last phone in the string, our final rule resembles this:

(3) null → [t] / X [ɪ] [-voice] _ #

If we find a third word pair that shows the same null → [t] change but has a different vowel in place of [ɪ], we then generalize (3) with the rule that describes the third pair, creating another hypothetical rule that replaces [ɪ] with the feature [+vowel]. The generalization continues like this for all word pairs in the set.

However, this process has the potential to generate an overabundance of rules that can seriously use up memory and slow down the generation process, particularly if one chooses to generate multiple potential rules per word set in order to determine which one is strongest and most consistent (Albright 2006). While the original learner did not state a need to delete potential rules, it also seemed to only generate one new rule per comparison. If, in the search for rules describing both morphophonological changes and resultant derived environment changes, we make at least two rules per generation, we run the risk of creating too many rules that come to the same conclusion, but with some crucial difference that causes minimal generalization to skip combining these rules. For example, if we consider the process of palatalization, this is the difference between creating two separate rules with both [i] and [ɨ] as the 'default' vowel, which will be elaborated on in the next section. Further, having the ability to remove underperforming rules also allows users to end generation processes without analyzing all the rules if it will only generate redundant data - Albright and Hayes also used this process to end the program when enough robust, generalized rules were created that there were no rules removed at the

checkpoint. This, while focusing less on the generation of rules, in fact helps keep the algorithm efficient by maintaining the rule set so that only the necessary rules are added and kept.

Similarly, over-generalizing runs the possibility of detracting from the richness of grammar. This, however, can also ease the process of determining possible final forms because it can more easily find rules that have similar structural properties to the currently-analyzed root; the less of a direct comparison between rule and root, the less likely it is that this rule will be chosen for the final form because the connection between the two appear more vague. The solution is to allow rules of varying levels of generalization (Albright and Hayes 1999), so that the narrow-scope generalizations can help develop hypotheses with higher effectiveness and confidence due to higher specificity.

2.3 Hypothesis Growth and Selection

Finally, the last principle of hypothesis growth stipulates that for any word and intended inflection, all potential surface forms are generated with the intent that at least one of these forms is valid. Then, using the generalized rules, these forms are ranked by likelihood of it to determine the likelihood of each one being the optimal change - from here, the most likely change is selected as the expected surface form. After this, the chosen form is compared against the 'real' surface form to see how similar or dissimilar the two are and whether new rules need to be derived as a result.

This process can help find the default mapping, or the default regular change for the particular conjugation or declension being tested. For example, in English present to past tense verbs, this change is X → Xd; though other changes exist, this is the most obvious and the one most likely picked when tested on an unfamiliar word. In fact, the Albright-Hayes algorithm had an original structure that tried to utilize the 'most likely' affix, but it was deemed too fragile

because it failed to determine irregular verb forms; because it searched for the most common affix mapping, which always evaluated to the default, every word was conjugated the same way. Thus, the key in making this model work is teaching it to "trust" any stem that is not the default, so that it is able to judge the reliability of non-default affixes.

The manner by which this mostly-likely form is chosen depends on an algorithm determining confidence in the choice (Mikheev 1997). Essentially, every form generated in the hypothesis set is weighed against the current rule database in order to measure how well it adheres to the rules - the better it follows these rules, the higher its confidence is ranked. One way of measuring is through the estimated proportion of success $\hat{p}$ (Mikheev 1997), seen in (4).

(4) $\hat{p} = \frac{x{:}number\ of\ successful\ guesses}{n{:}total\ number\ of\ compatible\ word\ tokens}$.

Where Mikheev deals primarily with guessing word classes, a success in Albright and Hayes' case would be the number of successfully generated past tense forms - that is, from the generated hypothesis set explained above, the chosen guess with the highest confidence matches the expected form. However, as Mikheev points out, this method allows for a large margin of error due to a lack of training data; his solution is to, in turn, calculate the lower confidence limit π_L. Albright and Hayes modify this system slightly to avoid 0s in numerators or denominators, but overall use a similar set of formulas. First, $\hat{p}$ is calculated using the formula in (5), where x_i and n_i are each instance of x and n defined above. Then, the lower confidence limit is calculated using the equation in (6).

(5) $\hat{p} = \frac{(xi + 0.5)}{(ni + 1.0)}$

(6) $\pi_L = \hat{p} - z_{(1-\alpha)/2} * \sqrt{\frac{\hat{p} * (1 - \hat{p})}{n}}$

In (6), the square root $\sqrt{\frac{\hat{p} * (1 - \hat{p})}{n}}$ is what is known as the 'estimate of true variance'. From these formulas (and, in Mikheev's case, his unmodified originals), it is determined that the most appropriate confidence limit is about 75%, which results in very low confidence for instances with a small scope (where scope is the number of cases) and more medium confidence for high scope. This prevents instances where the reliability (instances of correct mapping over instances of mapping in general) of a very small scope is high than that of a very large scope - for example, given a reliability value of 1.000 for 5/5 matches and .99 for 990/1000 matches, though the first has a higher value, the second should be more impressive because of the larger scope and breadth of data taken into account.

2.4 Albright-Hayes Algorithm in Practice

Given these principles, the learner is then fed a dataset to generate the grammar, so that it knows what to use for comparisons - given a pair of words, the first word is divided into PAQ and the second into PBQ, where P and Q are shared segments between both words and A and B are the site of morphological change. Either P or Q can be null; in the case of additive affixes, rather than morphology that changes a segment of a word (*string* to *strung*, for example), A can also be null. Note that, while the structure of the algorithm does allow for B to be null as well, this is likely counterintuitive, as the purpose of the algorithm is to find additive or transformational changes. While possible, Albright and Hayes did not test for the possibility of a null B, and thus this case will be ignored. For example, given the word pairs *string/strung* and *miss/missed*, we see the resulting parsing in figure 1 (Albright and Hayes, 1999). Note that, for *miss/missed*, Q and A are both null, and B is [t] - because the suffix is added to the word end, P is the entirety of the word before applying the suffix, and because the change is additive rather

than changing preexisting phones, A has no content.

P	Q	A	B
[mɪs]	null	Null	[t]
[kɪk]	null	Null	[t]
[str]	[ŋg]	[ɪ]	[ʌ]

Figure 1: breakdown of *miss/missed, kick/kicked,* and *string/strung* into its PAQ/PBQ segments

PSharedSegments	PSharedFeatures	QSharedSegments	QSharedFeatures
[mɪs]	null	null	null
[kɪk]	null	null	null
[str]	null	[ŋg]	null

Figure 2: further breakdown from figure 1.

A parser then breaks each word set down into smaller segments to compare against existing rules. First, the parser scans leftward from the beginning to find the place of change A so that it can evaluate the contents of P. Having found the beginning of A, the parser steps back one place at a time to find the longest segment of phones that the two words share, which is placed into a variable known as PSharedSegments. If there still exists some preceding phones that are not shared but have some shared features, these features are collected into PSharedFeatures, another variable. Whatever is left over in the P segment is considered residue and replaced with a variable, after which the same process is executed in reverse order on the Q segment, while retaining the change A→ B as the core of the rule. From the parsing in figures 1 and 2, we generate the rules found in (7) and (8).

(7) null → [t] / [mɪs] _ #

(8) null → [t] / [kɪk] _ #

We also generate another rule from the parsing of *string/strung,* shown in (9):

(9) [ɪ] → [ʌ] / [str] _ [ŋg]

This is, in effect, an elaboration on how the minimally generalized rules are generated, while taking into account what is compared between an existing rule and a new set that is added to this ruleset.

This is the point where the generalizations are key - in combining these rules, as seen in our minimal generalization in (3) above, the strings PSharedSegments and QSharedSegments must be robust enough to make an educated rule. Though rules may hinge on a single feature, such as word-final voicing, having this larger rule set does, as stated, allow the algorithm to pick hypotheses with higher confidence. The key is at this stage, when comparing shared segments between words, in deciding whether the ruleset would benefit from generalization. The various costs of each angle have been described, and as multiple rules are generated per word pair, this problem will only be exacerbated as more memory is used more quickly.

Here, during our division of word pairs into their various PAQ/PBQ segments, is where changes due to derived environments complicate matters - it becomes increasingly more difficult to subdivide a word into what is the place of change at the time of affix application versus what is influenced by this derived environment. Using data elaborated on in the next section, the Albright-Hayes algorithm would take a word pair in Polish - [krɔk] ('step') and [krɔt͡ʃɨt͡ʃ] ('to step') - and likely determine a rule that strips the final consonant from the root, treating [t͡ʃɨt͡ʃ] as the entire suffix. This would, in fact, capture the change in consonant, but a problem arises because a relationship between the affricate [t͡ʃ] and the vowel [ɨ] is overlooked. This is due to the process of palatalization, which is a prevalent change in Polish that occurs specifically in relation to the vowels [i] and [ɨ], and which here appears to be triggering a change from the consonant [k] to the affricate. Palatalization will be explained in further detail in the next section

What needs to occur, then, is some process that breaks the B in PBQ into further segments to determine what is, in a sense, B' or purely the affix, what is B_P (the affected zone that is truly an altered subset of P that has been affected by the derived environment), and what is B_Q (the same, but as a subset of Q). One simplified method of doing this is to break down the word by phoneme, as we have done before, but to create a mediary underlying form (Zeldes 2007). This abstracts a potential suffix to unbind the word from the final phonology, allowing us to create the rule for the suffix before looking at how the words change together when the suffix is applied. The problem with this approach, however, is that it requires some previous knowledge of phonological rules before beginning in order to create these abstractions, but is one method of building rules later on once rules are introduced to the learner. Because one of the main principles of this algorithm involves retaining previous hypotheses, the algorithm can relearn rules through generalization once the necessary palatalization structures have been discovered. This method, however, is extremely costly and could massively increase the runtime of the algorithm, depending on the number of rules that need to be re-evaluated at this point; instead, we must search for some system that is able to recognize an underlying form based on the root form of the word, mapping the modified result to the original so that they can be intrinsically linked when generating rules.

2.5 Limitations due to derived environments

Derived environments are occasions where the root is phonologically sound and the affix is phonologically sound but, when the affix is applied, it breaks some crucial phonotactic constraint in a language. This is fixed by modifying not only the affix, as is done in the rules tracked by the Albright-Hayes algorithm, but modifying the root word itself as well. As it stands, this algorithm would count this derived change as simply part of the affix; recall that in *string*

and *strung*, the areas of change A and B are [ɪ] and [ʌ], rather than either being left blank, meaning that situations where a morpheme 'overrides' another in this way simply has the original in one segment and the replacing morpheme in the other. However, it is crucial to note that this will result in one of two possibilities. In the first, the algorithm will under-generalize and will treat each derived environment as a separate rule - the affix is considered, but every surrounding change becomes a rule in and of itself as well. For example, the data in figure 3 shows three instances where applying one rule triggers a second to also be enacted. Though the same affix is added to all three instances, the resultant rules would categorize each word separately because of phonological changes that cross morpheme boundaries. In the second possibility, the algorithm will instead over-generalize and treat the affix and the derived environment as two parts of the same rule; in this case, the addition of the suffix and the change in vowel is captured, but the change in consonant would be reduced to some singular feature. Of these two possibilities, it is more likely to under-generalize and create overly specific rules; because of methods of combining phonological rules to generalize them (Albright and Hayes 2002), each individual rule would consider the derived-environment change as part of the entire phonological rule and be added as part of the affix rule, even though the affix is applied separately.

3. Background Survey

To determine how to counter the limitations caused by the algorithm's inability to account for derived environments, we will examine one particular example encountered in Polish. By modifying this algorithm to account for processes such as palatalization, more concise rule sets can be created, but special care must be taken to handle alternations of phones cautiously.

3.1 Palatalization in Polish

Final gloss	Root word	Root gloss	Suffix	Result
'to step'	kro[k]	'step'	[i]ć	kro[t͡ʃɨ]ć
'to frighten'	stra[x]	'fright'	[i]ć	stra[ʃɨ]ć[2]
'to weigh'	va[g]	'weight'	[i]ć	va[ʒɨ]ć

Figure 3: examples of derived environments in Polish (Inkelas 2014), modified from original transcription to better fit Polish phonology used elsewhere in this paper (Gussman 2007). Words here are written orthographically except where bracketed, in which case they are phonological and show the site of change.

In the data set provided in figure 3, this environment is created not from present to past tense verbs, but rather when taking a noun and turning it into a verb - the suffix *-ić* (pronounced as [iɕ] [Gussman 2007]) is one that denotes an unconjugated verb. This makes the derived environment very plain to see, and demonstrates that it is a change in the root word rather than applying an inflected suffix. Using Albright and Hayes' algorithm and assuming the rules will undergeneralize, our result rules would resemble those in (9) - (11).

(9) [k] → [t͡ʃɨt͡ʃ] / X [k] _#

(10) [x] → [ʃɨt͡ʃ] / X [x] _#

(11) [g] → [ʒɨt͡ʃ] / X [g] _#

[2] There is some debate about the nature of the affricate used here and the fricative in the next data point, specifically as to whether these phones are post-alveolar or retroflex. For the purpose of this paper, they will be listed as post-alveolar (Gussman 2007).

By Albright and Hayes' algorithm, this is a valid rule set; and indeed, a model that has stored these three rules will continue to generate verb forms for words that follow this pattern. However, these rules describe extremely similar processes: to turn a noun ending in a stop into a verb, we replace the stop with a new ending comprised of the palatalized form of the original consonant and the segment [ɨt͡ɕ].

Non-palatalized	**Gloss**	**Palatalized**	**Gloss**
[bawɨ]	'they (fem) feared'	[bʲawɨ]	'white'
[fɔk]	'seal'	[fʲɔk]	'hair lock'
[vara]	[beware]	[vʲara]	'faith'

Figure 4: further examples of contrastive palatalization in Polish. Data from Gussman 2007.

Palatalization is a massive phenomenon in many Slavic languages, and in Polish, it comes with a number of key features. Most notably, palatalization is used for contrast in Polish, as in the difference between words like [pasɛk], 'belt', and [pʲasɛk], 'sand', as well as other pairs seen in figure 4. This palatalization also controls the vowels that follow a palatalized consonant; speakers will often note the vowels [i] and [ɨ] as distinct and separate units, despite not considering the same difference between a palatalized labial and its non-palatalized counterpart (Gussman 2007). Most easily, one can say that [i] is only present after palatalized consonants, while [ɨ] never appears after palatalized labials such as [pʲ]. This is consistent with the changes seen in the above data - in each case, the final consonant is suitably palatalized, but because none of them are labials, [ɨ] replaces [i] as the vowels.

Please note, however, that in Polish, palatalization does not necessitate actually being a palatal consonant; alveolar fricatives and affricates, post-alveolar fricatives and affricates, and the liquid [l] can be used to denote palatalization (Gussman 2007). Further, though palatalization is fairly standard, there are some rules that allow a phone to alternate to a choice of palatalized

allophones. For example, [x] can palatalize to both [ʃ] and [ɕ], depending on context - [muxa] (nominative case for 'fly', as in the insect) becomes [muʃɛ] in the dative locative, but [vwɔx] (the adjective 'Italian') alternates to [vwɔɕi] in the nominative plural form. Similarly, some allophones also derive from the same phoneme, such as [s] as well as [x] being able to change to [ɕ] in some situations, such as the change from nominative to dative locative: [lɔs] (nominative case for 'fate') alternates to [lɔɕɛ] (Gussman 2007).

3.2 Testing Alternations

In both of these cases, the transformations must be kept distinct from one another; rules denoting the different changes allow for a richness in the language that facilitates such varied interplay between phonology and morphology (Zeldes 2007). Zeldes, in describing an algorithm that translates orthography to phonology, creates a system that tokenizes affixes to distinguish between those that are 'homographic' but 'morphophonologically distinct' - that is, affixes that are orthographically similar but whose meanings are different. As an example, the word 'siać' ('to sow') has the same ending as 'wypuszczać' ('to let out'). However, 'siać' is a perfective verb, while 'wypuszczać' is imperfective - in the second case, the suffix '-ać' is used because of a derivation from the perfective form of the verb, 'wypuścić'. Zeldes marks the suffix used in 'siać' as 'R2' (fully, /R2ać#/, where R2 specifies which 'class' the suffix is, 'ać' is the suffix being handled, and '#' only denotes that the suffix is word-final), while the corresponding suffix is 'R3' (or, similarly, /R3ać#/) (Zeldes 2007). Though not the case here, we see above how phones are able to alternate depending on the meaning of the affix. The Albright-Hayes algorithm, however, has no method of tokenizing morphemes to distinguish rules based on meaning, so only one manner of transformation - whether present to past tense verbs or nominative declension to locative - can be tested at a time.

Alternations in palatalizations, then, complicate the determination of the underlying forms. In one case, we default to [i] unless it follows any consonant that is not a palatalized labial; this is supported in that words can also begin with [i], and [i] also stands on its own as a word with meaning ('*and*'). Conversely, we can claim that [ɨ] is the default, with rules stating to alternate when after palatalized labial or at the beginning of a word. Either case explains the relationship between the vowels equally well, but the fact that both encompass such a wide subset of the language complicates any decision to favor one rule over another. Gussman also suggests the possibility that neither vowel is allophonous of the other; rather, the two vowels are categorically separate, but deficient in a way that there are certain places they cannot appear, such as the beginning of words or after non-palatalized consonants. Though entirely valid, this option does nonetheless ignore the relationship between the two vowels and their distribution. Due to this significant flaw, it is preferred to choose one vowel as the underlying form of the other, however arbitrarily; though the reasoning for either rule dominating the other may be weak, it establishes that the two rules are related nonetheless.

In the context of the Albright-Hayes algorithm, this does point out a serious issue to be considered when moving forward to integrate derived environments and cross-morpheme phonological changes. In the original algorithm, rule-creation was relatively simple, because it would be a change from the original A to the inflected B. In addition to this original rule, we must now determine how to manage rules that arise as we create these environments. For our data points above, do we change the vowel in the suffix because of the final consonant of the root, or do we change the root because of the vowel? If we later encounter the same affix added to a word in a different phonetic environment, the rule we generate can affect whether we are able to generalize two appearances of one affix into a single rule. Gussman's analyses offer

insight into the three ways we can handle the situation: write a rule in favor of the root, write a rule in favor of the affix, or consider each case independent and somewhat deficient. The third method ignores both the reality of allophony in most languages, where one phone is an underlying form of another. Because the Albright-Hayes algorithm and my proposed modifications work under the assumption that the surrounding features of a phone affect its surface form, the third proposal will thus will be ignored.

However, regarding the rule sets, in cases such as this vowel alternation where no one vowel is easily distinguishable as an allophone of another, it may in fact prove beneficial to retain both rules (that is, the rule favoring the affix and the rule favoring the root) as long as necessary. As rules are changed depending on input, any contradictions will be eliminated from the final set, and though some rules may be redundant, it is more likely that certain rules will prove more robust than others, being formed by more input and more generalized connections. Periodically, these less robust rules may be cut from the hypothesis as they fail to prove their repeated validity; otherwise, they take up necessary memory, and an excess of rules can also significantly slow down generation processes (Albright 2006). This will allow the learner to focus on the most robust rules by removing the under-generalized, redundant rules that occur only a handful of times and do not best describe the relationship between two phones.

Because palatalization is such an intricate set of phonological changes, it allows for a wide range of rules to be generated. Obviously, not all rules can be captured under this title, but understanding palatalization gives massive insight for how many morphophonological changes are executed in Polish. Namely, every conjugation or declension in Polish has some set of possible suffixes available to words depending on their phonology, as many languages do. Derived environments only occur when material from two different morphemes combine to meet

some criteria, breaking some rule that must then be rectified (Inkelas 2007). Palatalization, thus, presents a good method of studying these changes because they occur consistently across a wide range of declensions and conjugations, and learners tracking palatalization in Polish can be trained with massive amounts of data to determine these rules and, specifically, what is part of the affix and what is actually a change in the morpheme due to the affix.

4. Extensions and Modifications

To begin altering the Albright-Hayes algorithm, we must look more deeply into the structures we plan to analyze. The focus here is on verbs derived from noun forms in Polish, so that we have a consistent pattern to analyze. This also isolates the number of changes that can be observed - if we only track this one change, then generally, we only have to determine the identity of the suffix being added and what change it triggers in the root word as well. This falls in line with Albright and Hayes' original line of research - they trained their original model, in one instance, specifically on English present and past tense so it would track a consistent phenomenon. However, despite focusing on one particular phenomenon, we will nonetheless try to expand this theoretical basis as widely as possible, despite only testing for palatalization; this will give a more robust end result, giving a balanced algorithm that will more easily translate to multiple scenarios without losing sight of the original problem.

So, given a noun and verb pair in Polish, we can simulate how the model would read these two words to generate one rule in its original form. Suppose we take the word pair [pamʲɛⁿt͡ʃ] (*pamięć,* 'memory') and [pamʲɛⁿtat͡ʃ] (*pamiętać,* 'to remember'). First, we scan both words to find the segments in PAQ and PBQ. Finding P is simple: because it's everything in the root preceding the site of change, it is, in this instance, the string [pamʲɛⁿ]. However, dividing A/B from Q is where we meet a roadblock - we can view the change in these words, in the context of this specific algorithm, as either a replacement of the final consonant by the entire suffix string, or as an insertion of a shorter string without any removal. That is, we have one reading where Q is an empty string, A is the ending [t͡ʃ], and B is the ending [tat͡ʃ] - we remove the final consonant from the word and replace it in its entirety, leaving nothing after the place of change. Alternatively, we read Q as [t͡ʃ], A is null, and B is [ta] - we insert B into an uninhabited

spot in the root word, splitting it into two parts P and Q. While neither fully explains the processes being exhibited, the second reading would score low because word pairs with similar patterns would be difficult to find, since this particular parsing is more or less a coincidence for this word pair.

As we've shown, in our conflicting options for A and B, the model is unaware as to whether to use [tat͡ʃ] as a suffix or [ta] as an infix. The model, in fact, has no real preference for either suggestion until one is given enough priority over the other by a significant number of similarly-matching word pairs. It begs the question why any inherent preference would be needed - either case will lead to a correct output, so is it necessary to go so in depth when the ouput that is attainable by the Albright-Hayes algorithm is sufficient? The concept of strict cyclicity (Kenstowicz 1994) gives us a hierarchy that does give merit to this subdivision. These cyclic rules describe a sequence of rule applications where one derivation follows another. To enforce this idea of cyclicity, we need to treat this palatalization as a process that can be repeated across other strings of phones. The explanation using only an infix-based rule is sufficient, but it does not explain what is actually occurring in the change. If [pamʲɛⁿtat͡ʃ] is conjugated with the belief that [ta] is an infix, then in forms such as [pamʲɛⁿtam] ('I remember'), it comes to be assumed that the string [ta] is the marker that this word is a verb and [m] marks it as first person singular present, which is incorrect. For these words to be executed in the proper cyclical order, the suffix must be retained as a whole rather than broken down into arbitrary pieces.

Taking this reading of the word pair, where A and B are the entire word endings, we then must continue subdividing B to find the residue of A, which we will call AResidue. Finding the exact method is difficult - if we merely scan B for the set of phones with the highest match to A, then we essentially return to the problem of assuming insertion. Both A and B contain the phone

[t͡ʃ], so BResidue will be the final consonant in [tat͡ʃ], marking this as the 'leftover' of A and marking the 'true' affix as [ta]. Our goal is to match [t͡ʃ] to the palatalized [t] in B, and then noting that the addition of the suffix [at͡ʃ] is what triggers this change in [t͡ʃ].

The easiest way to do this, given that our particular focus is on suffixes, is to eliminate the existence of Q altogether, and instead divide B into two parts, B_P and B_A - if A remains the segment of the root that changes due to affix application, B_P is the part of the root that is altered (in this case, palatalized), while B_A is the affix and its resulting change, excluding any part of the root. This would linearly scan through each pair and set each post-affix word to a pattern of P-B_A-B_P, which suits this situation without much problem. This however limits some of the universality of the original algorithm. The new process would not be limited to only palatalization in Polish noun-to-verb morphology, as the algorithm would be able to work on any suffix-modified word pair, and in cases where no part of the root is modified, B_A evaluates to null; in fact, it could also be modified to allow for reading for prefixes as well, where if B precedes P, then B_A appears on the side closer to P. Aside from some limited improved efficiency, a modification like this does not significantly improve the algorithm, and thus universality will be prioritized.

However, issues arrive when dealing with infixes. As we see with words like [pamʲɛⁿtat͡ʃ], a prefix or suffix can easily be confused with an infix, if that infix contains part of the original word. The original algorithm accounted for this by generalizing across any of these such instances, where if there was one confusing case like this out of dozens or hundreds of word pairs, the instances where it was correctly recognized as a suffix are given more weight until the incorrect inference is given lower confidence and ignored. In turning this algorithm into a multi-part process that breaks B into smaller segments, we not only inherit some of the inherent issues

with infixes in this situation, but we exacerbate the problem by branching into even more possibilities, allowing the root word to leak into the affix on either end. We can try to remedy this by changing the substructure within B to actually match the superstructure - we have a sequence B_P, B_A, and B_Q, where B_P receives information from P, B_Q receives information from Q, and B_A is untouched, new information added by an outside affix. This allow us to pinpoint exactly where in the affix we have created a derived environment, after which we can determine the change it causes.

Unfortunately, from a single word, there is no way to determine whether it is better to treat this affix as an infix or a suffix. Continuing to use [pamʲɛⁿtat͡ʃ] as an example, we can segment this word in two different ways, shown in figure 5. Both of these are equally valid when treated as an isolated case. Because of this, we cannot throw away either representation at this step; rather, as we continue parsing word pairs, we must continue to retain both versions of the rule until one has been generalized and applied to enough other pairs that it can sufficiently outweigh the other. This is where our minimal rule generalization becomes key, given the multitude of rules generated per iteration, and why an effective clean-up system to reduce the number of applicable rules can crucially change the outcome and speed of the algorithm when it runs.

P	**B_P**	**B_A**	**B_Q**	**Q**
[pamʲɛⁿ]	[t]	[at͡ʃ]	∅	∅
[pamʲɛⁿ]	∅	[ta]	∅	[t͡ʃ]

Figure 5: Parsing of [pamʲɛⁿtat͡ʃ], first using [at͡ʃ] as a suffix, then using [ta] as an infix.

In fact, the rule generation will not end here. Up to this point, we have only determined which segments have been changed between the original word phrase and the inflected form, by first determining the entire change, and then breaking this section down into changes categorized

by origin, either the root or the affix. In the vein of the first version of the algorithm, we must also generate rules to describe both of these phenomena, and in doing so, we have to describe both the affix and how the sub-segments of B change.

The first question in this step is how to organize the rules. Because derived environments are only found after the application of the rule, 'deriving' from where the root and affix meet, we must first describe the form of the underlying affix so that we understand what is truly being changed. Once we determine what the contents of B_Q and B_P are, we remove these segments from B_A to describe the affix itself and build a rule in the original format based on this, using the segments from A transformed into B_A. Thus, for the two versions of [pamʲɛⁿtat͡ʃ], we would generate the rules in (12) and (13).

(12) [t͡ʃ] → [t͡ʃat͡ʃ] / [pamʲɛⁿ] _#

(13) [∅] → [ta] / [pamʲɛⁿ] _ [t͡ʃ]

The second rule, naturally, will fully describe the change in a manner appropriate to Albright and Hayes' original algorithm. However, in the case of the first, the result better resembles [pamʲɛⁿt͡ʃat͡ʃ] and does not account for the palatalization. This is where we encounter our derived environment - the features in B and A matched, so we determine that some part of the root changed as a result of the addition of the affix.

As a follow-up to this rule, we then create a second rule that, when given the output of our affix-based rule, can search for this environment and change our rule. Similarly to our original set of rules, this again starts at the most generalized form and builds outward based on generality, so that we can retain as few rules as possible. The secondary rule, as applied to (12), would resemble the rule shown below:

(12.1) [t͡ʃ] → [t] / [pamʲɛⁿ] _ [at͡ʃ]

This, then, will specify that only the first instance of [t͡ʃ] is modified, but will still give us our desired output of [pamʲɛⁿt͡ʃat͡ʃ]. As we introduce other words into our dataset, rules generated from these sets will be generalized with those we have already created.

P	**A**	**B**	**B_P**	**B_A**	**B_Q**	**Q**
[kɔrzɨ]	[ʃt͡ʃ]	[stat͡ʃ]	[st]	[at͡ʃ]	∅	∅

Figure 6: parsing of [kɔrzɨʃt͡ʃ] to [kɔrzɨstat͡ʃ] as an example

Let us take a second example so that we can see the rule addition in progress. For this second pair, we will use [kɔrzɨʃt͡ʃ] (*korzyść,* 'profit') and [kɔrzɨstat͡ʃ] (*korzystać*, 'to profit'), and we will demonstrate only the suffix rules. For our noun form of the suffix rule, we use the parsing laid out in figure 6 to show how A → B between P and Q, remembering that B_P, B_A and B_Q are subsections of B. Thus, our primary rule comes out to:

(14) [ʃt͡ʃ] → [ʃt͡ʃat͡ʃ] / [kɔrzɨ] _ #

This primary rule is fairly similar to the one produced in (10), with the exception of a secondary vowel. If we apply the method of derived environment rule generation as we have before, we acquire this rule as a result:

(14.1) [ʃt͡ʃ] → [st] / [kɔrzɨ] _ [at͡ʃ]

This, however, also does not fully describe the phenomenon at play, which leaves us at another crossroads: do we retain this rule as it is, or do we generate multiple sub-rules from increasingly smaller segments of B?

One option, then, is to recognize that B_P is made up of multiple phones and to map them individually to segments of A - that is, we map [t͡ʃ] to [t], as previously done, but we also map [ʃ] to [s]. By aligning these phones in this way, we add more rules that continue to describe how these changes can be made. Thus, in place of (14.1), we instead get the two following rules:

(14.2) [t͡ʃ] → [t] / [kɔrzɨʃ] _ [at͡ʃ]

(14.3) [ʃ] → [s] / [kɔrzɨʃ] _ [t͡ʃat͡ʃ]

This, in effect, gives us a more nuanced view of how palatalization affects Polish by cascading from the phone closest to the affix outward. We specifically choose this ordering, changing [t͡ʃ] before [ʃ], because of its proximity to the suffix - otherwise, it would 'block' [ʃ] from changing accordingly. There are no nouns ending in [ʃ] that alternate to [s] in the verb form, so choosing that ordering first would be nonsensical. We could easily retain the rule in (12.1), but these resulting rules give more depth to other rules we generate. In addition, this also upholds the concept of strict cyclicity by allowing for multiple changes to occur as a result of one another, where the ordering of these rule-applications can affect the final output.

Having achieved these two sub-rules, we then add (14.2) to (12.1), in the minimal generalization style described earlier. The mapping is identical, so we are able to match these two rules together for generalization. The sequence following our site of transformation is also identical, so there is no generalization needed here. However, the sequences before our sites of mapping, [pamʲɛⁿ] and [kɔrzɨʃ], do not share any relevant features. Both of these preceding sequences can thus be replaced with the variable X, so if any future words come along that take on the [at͡ʃ] ending, they can be inflected properly in this form; similarly, if we have any other words like [kɔrzɨʃt͡ʃ], this further enables us to maintain the multi-part segment subdivision while controlling the number of rules we create.

Suppose, then, that these two rules are now a complete trained set for our algorithm in Polish. We can move on to the generation phase by giving the model a string to conjugate as it sees fit. For example, we can now provide the string [kat͡ʃ], which is not a word in Polish, and expect it to be palatalized. Due to the principle of hypothesis growth, we build a set of all possible rule applications - in our case, either an infix or a suffix. Generation using our primary

rules would give us a result of either [katat͡ʃ] using the infix rule, or [kat͡ʃat͡ʃ], using the suffix. Once we apply our secondary rule, then [kat͡ʃat͡ʃ] also resemble [katat͡ʃ], because we scan through the string to find this conflicting [t͡ʃa] environment, which hitherto has not been legal.

Though the infix rule does generate the 'correct', expected string in fewer steps, it must be noted that the rules generated still do not uphold ideas of the Polish language previously known, and thus has no basis in reality. We know, from verb data presented here, that verbs are denoted by the application of a suffix ending - for example, the suffix [it͡ʃ] in the verbs presented in figure 3. If we were to print out a final summation of all the rules presented in Polish, it is necessary for these extra distinctions to be made to uphold these patterns in how languages construct words of certain types or inflect in certain ways. Further, we learn more if we allow for this kind of alternation, as palatalization is something that occurs outside of the singular context of this noun-to-verb change - if we treat the entire segment as simply part of the affix, we accept that, in the language being evaluated, certain phones never appear in the same context after adding some affix by sheer coincidence, rather than acknowledging that the phone instead is actively changed between due to the addition. To this end, though this alternative algorithm in some ways overcomplicates the sufficient system previously created, it does add some nuance that better explains why these patterns act the ways they do.

What cannot be described here directly, however, is what Albright and Hayes observed in their preliminary trials, in speakers of Polish creating 'new' derived environments. The outcome of these trials were not specified in the original paper or alongside the files for the algorithm itself, where the training data sets were listed. However, it is likely that these results can theoretically be replicated in the rule-generalization process is fine-tuned in such a way that the rules are able to generalize by feature patterns rather than entire strings. This changes the

structure of the algorithm in how the phones are rendered, whether as specific strings or characters or rather as collections of sets of features. If altered in this way, the rules can be diminished to the specific features that constitute the changes in the derived environments, elegantly generalized rather than split into multiple rules describing similar changes. Thus, this case should be accounted for.

The difficulty here is that palatalization may not be realized in every phone in necessarily the same pattern. Regardless, the new rule format will allow for this particular aspect of these trials to be realized, capturing palatalization-style changes by matching similar features in unfamiliar environments. Specifically, it allows for the possibility that, when certain groups of phones share particular attributes, features that manifest as a result of these shared attributes can be tracked so that other phones can also join this group. For example, regarding palatalization, phones are generally palatalized when followed by the vowel [i] (Gussman 2007) - this is represented in the orthography as well, where the string 'si' is rendered in speech as [ʃ], without a vowel component, due to the vowel [i] marking it as palatalized. Palatalized consonants (and vowels) are a large group of phones modified by this or similar processes, so phones modified in this way must share similar features inherited by the [i].

To summarize, the main extension to this algorithm is to detect derived environment changes by continuously breaking down segments that denote change, making reference to the original place of change to detect underlying forms of mapped phones. In our substructure B_P, B_A, and B_Q, we isolate the affix in one segment of this entire area of change, then recursively take phones off the other segments in B while we generate rules, showing how these phonological changes build off one another to attain the final product from the initial root-affix combination. This gives credence to the theory of derived environments, adding a new layer to a

pre-existing model that gives a deeper insight to how these final verb forms might appear thanks to the initial appearance of a root rather than the form of the suffix. However, to more fully support this theory, the process of rule-generalization outlined by Albright and Hayes needs to be expanded to allow for rules that depend on features rather than full phones. While the primary rules need to insert or replace a phone with some concrete morpheme, the secondary rules only modify what already exists. For these rules to be most successful, they can determine whether a certain phone within a morpheme can change depending on its feature and placement, not simply on its identity as a particular phone - by isolating the exact features that cause palatalization, we can account for newly-created environments based on similarly-structured rule forms.

5. Critique and Future Work

In the future, my main goal is to take this theoretical breakdown and apply it to Albright and Hayes' original algorithm, modifying the code in such a way that it is able to reflect the ideas presented here. In terms of the coding, this will complicate some of their initial structures by introducing a more heavily recursive version of the algorithm that continually looks for new broken 'rules' in the generated data set. Because it will repeatedly look over the generated traversal through these strings, looking for areas that can be modified by our generated rules, it will massively increase the complexity of this problem - every time it finds some phonotactic constraint broken, it will attempt to correct it using our generated set of secondary rules before searching again for newly-occurring errors. One way to curb this is by limiting the search to only the B segment of the generated phone, so these rules do not search for infractions elsewhere in the word rather than focusing on the suffix and its place of generation. This will hopefully allow for a natural break in the algorithm, barring any situations where two rules may contradict one another. Nonetheless, possible forms and possible rules will continually stack on each other in a way that may overload the memory stack, even if we maintain a finite number of rules and forms, due to sheer quantity. Thus, these rule and form sets must be handled carefully, with particular attention to 'cleaning up' rule sets by removing less relevant rules.

However, this can in turn complicate the concept of 'confidence' by continually changing word forms. Given Albright and Hayes' original intent for this algorithm, it would create a massive hypothesis base from every possible rule and pick the one that appears mostly likely. To integrate our new rule scheme into this same format, the difference between primary and secondary rules (here used arbitrarily to denote the difference between the rule describing the affix and the rule describing the change in the root as a result of a derived environment) would

have to be formalized and made more concrete, almost treating them like two different kinds of rules. Primary rules would have to be barred from being acted upon after the initial application, while secondary rules would specifically only be triggered after the application of a suffix.

This same repeated looping described above would complicate generation of hypothetical verbs by intentionally bringing them all closer to the theoretical 'expected' form - applying these secondary rules would likely increase the confidence of all forms by ensuring that they all adhere to at least a certain number of phonological rules. Conflicts with the root noun should nonetheless allow for certain results to be preferred over one another, but in the event that this is not the case, the threshold for confidence may need to be retooled to better represent a model that is actively changing generated forms to bring them closer to center.

This is also a flaw with using Polish in particular as a basis for this kind of problem. English was a suitable test language for the original language because of the high number of irregular forms; Polish, meanwhile, has significantly fewer irregular forms, at least in the case of noun to verb generation. The verb 'być' ('to be') is irregular, but in terms of verb conjugation, Polish also manages to stay relatively regular, though it has a significant number of verb endings, and not all verbs have a corresponding noun form to test. However, it is possible to test this algorithm on multiple aspects of Polish morphology - the number of verb endings still leaves some room for testing conjugation between forms. The difference between imperfective and perfective forms of verbs is also a candidate, given that there is a general regular pattern to the generation of perfective forms that nonetheless has a significant number of irregular forms. Declension of nouns can also be tested, especially given that some nouns can decline in similar ways across declensions and some cannot, depending on the gender of the noun. Approaching this problem from the single viewpoint of palatalization in noun-to-verb affixes may be too

narrow of a viewpoint and, when the algorithm is applied, some phenomena may fail to be detected because of the manner in which this was developed. Though I will continue to focus on this particular morphological change, it does not invalidate that changes may have to be made if I intend to apply this same algorithm to another group of words in another language.

Another complication is the event where these infix-based rules are not canceled out, instead being added to by the same word pairs as the main suffix-based rule due to a constant pattern. Eventually, this may turn into a case where, when we generate verb forms based on an input rather than parsing a data set, we pick randomly between the two possible forms. Realistically, if this is the case, then the difference between the two forms will likely be negligible; however, it complicates any intended model for a particular language with a known pattern structure. This begs the question of whether it will be realistic or appropriate to pre-program any sort of standards of a language without allowing the model to discover it on its own - if Polish does not allow for infixes in noun-verb generation, does the program fail if this must be specified to choose the 'correct' rule, even if the 'incorrect' rule generates a nonetheless identical form? If I choose not to go this route, what would instead constitute an appropriate 'clean up' method to get rid of rules deemed insufficient, if only on the basis of nuance?

6. Summary and Conclusion

Though Albright and Hayes created a robust algorithm for the general concept of morphophonological rule generation, they cited specific areas where further research and development would account for certain phenomena. Polish was one of these such languages because, though the algorithm could recognize segments where there was a phonological change, it was unable to look into the substructure of this change. This meant that derived environments, the changes that came about as a result of some morphological change breaking a phonotactic constraint and having to change the root, were not recognized because the algorithm read the altered section of the root as merely part of the affix itself. As shown here, this substructure can be constructed by detecting phone mappings between areas of change in the root and result, allowing rule sets to be built based on specific sections of this area rather than the area as a whole. In turn, this means that if, post-training, a root word is given to the model, the algorithm can take a modified word and reconfigure 'illegal' strings based on the phonology of that given language, altering the set of possible final results. This shows that there is a future for the computational modelling of derived environments, replicating how we as humans are able to consider the cyclical nature of these rules.

References

Albright, Adam, and Bruce Hayes. "An Automated Learner for Phonology and Morphology." University of California, Los Angeles. 1999.

Albright, Adam and Bruce Hayes. "Modeling English Past Tense Intuitions with Minimal Generalizations." Morphological and Phonological Learning: Proceeding of the 6th Workshop of the ACL Special Interest Group in Computational Phonology (SIGPHON), Philadelphia, July 2002, pp. 58-69. Association for Computational Linguistics.

Albright, Adam. "Modeling Productivity with the Gradual Learning Algorithm: The Problem of Accidentally Exceptionless Generalizations." Gradience in Grammar. Ed. Bruce Hayes. Oxford: Oxford UP, 2006.

Gussmann, Edmund. The Phonology of Polish. Oxford: Oxford UP, 2007.

Mikheev, Andrei. "Automatic Rule Induction for Unknown-Word Guessing." University of Edinburgh, 1997.

Inkelas, Sharon. The Interplay of Morphology and Phonology. Oxford: Oxford UP, 2014. Print.

Zeldes, Amir (2007) "Abstracting Suffixes: A Morphophonemic Approach to Polish Morphological Analysis". Zeitschrift für Sprachwissenschaft [=German Journal of Linguistics] 26(2), 347-370.