Finding regularity in Tlingit verb prefixes

Marcel K. Goh

Abstract. [Put an abstract here. Possibly think of a better title; the one above is something I came up in five seconds that more or less describes the mumbo-jumbo my program produces. A more ambitious program would have resulted in a title of "Automatic segmentation and glossing of Tlingit verbs". But alas my program does no such thing.]

1. Introduction

[Probably an intro paragraph describing the Tlingit language as a whole, where it fits into language families, and the current linguistic situation. Briefly mention dialectal issues. Lure the reader in. Mention related work, etc. References I will use are [2], [3], [4], [1], and [5], and some others as well.]

2. Sequences of states

[Formalise the problem. Describe how the format of [3] allows the tables to be decomposed into a dictionary that maps sequences of underlying morphemes to sequences of actual observed prefixes. We study the inverse problem. Given a sequence of prefixes, how do we identify the original morphemes that they represent? Describe what the tags mean and how they will help us check when our rewrite approach has succeeded.]

Strings and tags. Formally, our program deals with the monoid of all strings over a finite alphabet A (the set of all letters in the Tlingit orthography), with the binary operation of concatenation; strings in this set will be written in a fixed-width typeface to distinguish them from the ordinary text of the paper. As an exception to this convention, we use ϵ to denote the identity element (the empty string). The data found in [3] is labelled, in the sense that every string has been partitioned into substrings, each tagged with the type of prefix they represent. These tags are important to our program, so we will list all of them here.

i) A *disjunct prefix* is tagged with Q. This may be a qualifier prefix, an incorporated noun prefix, or an object prefix.
ii) The *irrealis prefix* is tagged with R. The only two possibilities here are $R(\texttt{u})$ and $R(\texttt{w})$.
iii) *Aspect prefixes* are tagged with A. An example is the perfective prefix, which can either be $A(\texttt{wu})$ or $A(\texttt{u})$.
iv) The *modality prefix* is tagged with M. The underlying form is always $M(\underline{\texttt{g}})$.
v) *Subject prefixes* are tagged with S.
vi) The passive, antipassive, or middle voice is indicated by a `d-` *prefix*, which is tagged with D.
vii) Any of the `s-`*,* `l-`*, or* `sh-` *prefixes* are classified under the F tag, and we only deal with the `s` case, though it is noted in [3] that the phonological patterns are more or less analogous in the other two cases.
viii) The stative `i`*-prefix* is given the I tag.
ix) *Epenthesis* is indicated by the E tag.

Our program makes crucial use of these tags to detect when a sequence of observed prefixes corresponds exactly to a sequence of morphemes. We denote by A^* the set of all words formed from letters in the orthography, and if the nine tags above are viewed as functions, we can let B be the union of images of $A^* \setminus \{\epsilon\}$ under each of the nine functions above. We will call the elements of B *states*; an example is $S(\texttt{yi})$. Note that we never tag the empty string ϵ. Next, we consider the be the set of all nonempty sequences of states, denoted B^+; for instance, the sequence $Q(\texttt{ka})S(\underline{\texttt{x}})D(\texttt{d})I(\texttt{i})$ is an element of this set. We will let

$S = B^+ \cup \{\epsilon\}$, where ϵ is not tagged. This identity element will be important later when we start defining rewrite rules as functions from S to itself.

Hidden and observed sequences. Although the data that the program uses is formatted as a table, it is simpler to think of it as a map (or dictionary) from a set $H \subseteq S$ of sequences of underlying morphemes to a set $K \subseteq S$ of actual observed sequences of prefixes. When sequences of tags are the same, it is clear which prefixes are represented by each part of the verb. For example, one of the entries in the dictionary is

$$Q(\texttt{ji})A(\texttt{wu})S(\texttt{du})D(\texttt{d})F(\texttt{s})I(\texttt{i}) \mapsto Q(\texttt{ji})A(\texttt{m})S(\texttt{du})D(\texttt{d})F(\texttt{z})I(\texttt{i})$$

and it is clear which substrings of the actual verb correspond to respective prefixes. We will say that such an entry is *resolved* and *unresolved* otherwise. An example of an unresolved entry is

$$Q(\texttt{du})R(\texttt{w})A(\texttt{g})M(\underline{\texttt{g}})S(\texttt{du}) \mapsto Q(\texttt{du})A(\texttt{g})E(\texttt{a})M(\underline{\texttt{x}})S(\texttt{du}),$$

where the correspondence between states is not immediately obvious. Looking at more entries in the dictionary, we see that the transformation

$$A(\texttt{g})E(\texttt{a})M(\underline{\texttt{x}}) \to R(\texttt{w})A(\texttt{g})M(\underline{\texttt{g}})$$

occurs a fair few times, so we might consider applying this transformation to every string in the dictionary and see if it causes more entries to be resolved. This suggests a semi-computational approach to finding regularity in the charts, for if we can cook up a relatively small set of transformations that seem to underlie all of the phonological and morphological irregularity in the Tlingit verb, we can use a computer program to verify if chart is covered by the rules.

3. Rewrite rules

For $x, y \in S$, $x \neq \epsilon$, we formally define a *transformation* $x \to y$ to be a function from S to itself that replaces the *first* instance of x in a word $w \in S$ with y. A *rewrite rule* is a finite sequence of transformations. Although transformations are formally functions and functions are applied from right to left, we will apply transformations in a rewrite rule from left to right, to preserve the sanity of the (presumably English-speaking) reader. So if T_1, T_2, and T_3 are transformations, the rewrite rule $f : S \to S$ given by $f = T_1 \circ T_2 \circ T_3$ will be written T_3, T_2, T_1. This section will enumerate the twenty-eight rewrite rules that we will use to resolve every entry in the dictionary obtained from the charts [3].

Epenthesis and irregular disjuncts. With the luxury of having tagged data, an obvious thing to do is to remove all instances of epenthesis in the dictionary, since, more or less by definition, epenthesis has no underlying meaning. So we would like a rule along the lines of $E(*) \to \epsilon$ repeated infinitely many times, where the star stands for any possible string, but this does not satisfy our requirements for a well-defined rewrite rule. Thankfully, in our data the only letters that are ever epenthesised are `a`, `i`, and `o`. In the case of `a`, there are at most four instances of epenthesis in a given word, and in the other two cases there is at most one instance, so the rewrite rule

i) $E(\texttt{a}) \to \epsilon, E(\texttt{a}) \to \epsilon, E(\texttt{a}) \to \epsilon, E(\texttt{a}) \to \epsilon, E(\texttt{i}) \to \epsilon, E(\texttt{o}) \to \epsilon$

successfully eliminates all epenthesis from the dictionary. Another rewrite rule that simply cleans up the data is the regularisation of certain irregular disjuncts:

ii) $Q(\texttt{e}) \to Q(\texttt{a}), Q(\texttt{k}) \to Q(\texttt{ka}), Q(\underline{\texttt{x}}') \to Q(\underline{\texttt{x}}'\texttt{e}), Q(\texttt{j}) \to Q(\texttt{ji}), Q(\texttt{t}) \to Q(\texttt{tu})$

Labialised velars and uvulars. When a velar or uvular stop or fricative is followed by a `w`, the underlying form often has a `wu` or `u` (or variants thereof) *preceding* the consonant. These represent the perfective aspect and irrealis mood, respectively. In the case of the subject $\underline{x}$, the two cases are handled by the rule

iii) $S(\underline{\texttt{x}})A(\texttt{w}) \to A(\texttt{wu})S(\underline{\texttt{x}}), S(\underline{\texttt{x}})R(\texttt{w}) \to R(\texttt{u})S(\underline{\texttt{x}}), S(\underline{\texttt{k}})R(\texttt{w}) \to R(\texttt{u})S(\underline{\texttt{k}})$.

Note that we didn't change $S(\underline{\texttt{k}})$ to $S(\underline{\texttt{x}})$ in the third transformation; this will be done later. In the case that the consonant indicates aspect or modality, the `w` can only indicate the irrealis mood:

iv) $A(\texttt{k})R(\texttt{w}) \to R(\texttt{u})A(\texttt{g}), A(\underline{\texttt{g}})R(\texttt{w}) \to R(\texttt{u})A(\underline{\texttt{g}})$

v) $M(\underline{\texttt{g}})R(\texttt{w}) \to R(\texttt{u})M(\underline{\texttt{g}})$

Elision. It is quite difficult to deal with elision using rewrite rules, since one has to guess where the missing state should be inserted. An example of this annoyance is the systematic elision of the `d-` prefix when it is followed by the `s-` prefix. A solitary `s-` prefix is indicated by $F(\texttt{s})E(\texttt{a})$, but since it is convenient to delete epenthesis early in the rewriting process, we are left with a situation where $F(\texttt{s})$ may stand for either $D(\texttt{d})F(\texttt{s})$ or $F(\texttt{s})$. Thus we have the rule

vi) $F(\texttt{s}) \to D(\texttt{d})F(\texttt{s})$,

as well as its "inverse" $D(\texttt{d})F(\texttt{s}) \to F(\texttt{s})$. Ambiguity due to elision also occurs with the prefixes $R(\texttt{u})$ and $A(\texttt{wu})$. We will defer a more general discussion of resolving this ambiguity by "chaining" rewrites to the next section.

Modality prefix. The modality prefix $M(\underline{\texttt{g}})$ causes various shuffling of prefixes, when it coincides with the aspect prefix $A\underline{g}$ and its variants. The first thing to do is to simplify the problem a bit by modifying the following two strings (but not their tags):

vii) $M(\underline{\texttt{x}}) \to M(\underline{\texttt{g}}), A(\texttt{k})M(\underline{\texttt{g}}) \to A(\texttt{g})M(\underline{\texttt{g}})$

This rule will not cause the further resolution of any entries, but reduces the number of transformations we need in the following few rules. The first of these is a rule that removes an irrealis prefix in the vicinity.

viii) $A(\texttt{g})M(\underline{\texttt{g}})R(\texttt{w}) \to A(\texttt{g})M(\underline{\texttt{g}}), A(\texttt{g})R(\texttt{w})M(\underline{\texttt{g}}) \to A(\texttt{g})M(\underline{\texttt{g}})$

This prefix may need to be added back later, but it will precede the aspect prefix:

ix) $A(\texttt{g})M(\underline{\texttt{g}}) \to R(\texttt{w})A(\texttt{g})M(\underline{\texttt{g}}), A(\underline{\texttt{g}})M(\underline{\texttt{g}}) \to R(\texttt{w})A(\underline{\texttt{g}})M(\underline{\texttt{g}})$

Next, we have two "cleanup" rules that handle the mess that happens when the modality prefix clashes with a $S(\underline{\texttt{x}})$ prefix:

x) $A(\texttt{k})S(\underline{\texttt{k}})R(\texttt{w}) \to R(\texttt{w})A(\texttt{g})M(\underline{\texttt{g}})S(\underline{\texttt{x}}), A(\texttt{k})R(\texttt{u})S(\underline{\texttt{k}}) \to R(\texttt{u})A(\texttt{g})M(\underline{\texttt{g}})S(\underline{\texttt{x}})$,
$A(\texttt{k})R(\texttt{w})S(\underline{\texttt{k}}) \to R(\texttt{w})A(\underline{\texttt{g}})M(\underline{\texttt{g}})S(\underline{\texttt{x}})$

xi) $A(\texttt{k})S(\underline{\texttt{k}}) \to A(\texttt{g})M(\underline{\texttt{g}})S(\underline{\texttt{x}}), A(\texttt{g})S(\underline{\texttt{k}}) \to A(\texttt{g})M(\underline{\texttt{g}})S(\underline{\texttt{x}}), A(\underline{\texttt{k}})S(\underline{\texttt{k}}) \to A(\underline{\texttt{g}})M(\underline{\texttt{g}})S(\underline{\texttt{x}})$

Velar aspect prefixes. The next few rules handle the irrealis prefix in the vicinity of a velar aspect prefix. In most cases the irrealis is elided, so we must guess where it originally stood. First we have the rule

xii) $A(\texttt{g})R(\texttt{o}) \to R(\texttt{u})A(\texttt{g})$,

and to make our life easier, we will also regularise some of the subject prefixes:

xiii) $S(\texttt{y}) \to S(\texttt{i}), S(\texttt{ee}) \to S(\texttt{i}), S(\texttt{yee}) \to S(\texttt{yi}), S(\texttt{ye}) \to S(\texttt{yi}), S(\texttt{too}) \to S(\texttt{tu}), S(\underline{\texttt{k}}) \to A(\underline{\texttt{g}})S(\underline{\texttt{x}})$

This allows us to systematically add $R(\texttt{u})$ before the aspect prefix:

xiv) $A(\texttt{k})S(\texttt{du}) \to R(\texttt{u})A(\texttt{g})S(\texttt{du}), A(\texttt{g})S(\texttt{du}) \to R(\texttt{u})A(\texttt{g})S(\texttt{du})$,
$A(\underline{\texttt{x}})S(\texttt{du}) \to R(\texttt{u})A(\underline{\texttt{g}})S(\texttt{du}), A(\underline{\texttt{g}})S(\texttt{du}) \to R(\texttt{u})A(\underline{\texttt{g}})S(\texttt{du})$

xv) $A(\texttt{g})S(\texttt{i}) \to R(\texttt{u})A(\texttt{g})S(\texttt{i})$

xvi) $A(\texttt{g})S(\texttt{yi}) \to R(\texttt{u})A(\texttt{g})S(\texttt{yi}), A(\underline{\texttt{x}})S(\texttt{yi}) \to R(\texttt{u})A(\underline{\texttt{g}})S(\texttt{yi}), A(\underline{\texttt{g}})S(\texttt{yi}) \to R(\texttt{u})A(\underline{\texttt{g}})S(\texttt{yi})$,
$A(\underline{\texttt{g}})S(\texttt{i}) \to R(\texttt{u})A(\underline{\texttt{g}})S(\texttt{yi}), A(\underline{\texttt{g}})S(\texttt{ee}) \to R(\texttt{u})A(\underline{\texttt{g}})S(\texttt{yi})$

xvii) $A(\texttt{k})S(\texttt{tu}) \to R(\texttt{u})A(\texttt{g})S(\texttt{tu}), A(\texttt{g})S(\texttt{tu}) \to R(\texttt{u})A(\texttt{g})S(\texttt{tu})$,
$A(\underline{\texttt{x}})S(\texttt{tu}) \to R(\texttt{u})A(\underline{\texttt{g}})S(\texttt{tu}), A(\underline{\texttt{g}})S(\texttt{tu}) \to R(\texttt{u})A(\underline{\texttt{g}})S(\texttt{tu})$,

Lastly, because the $\underline{\texttt{g}}\underline{\texttt{x}}$ pattern performs double duty to indicate both aspect and modality, we will need to a rule to switch from aspect prefixes to modality prefixes:

xviii) $A(\underline{\texttt{g}})S(\underline{\texttt{x}}) \to M(\underline{\texttt{g}})S(\underline{\texttt{x}})$

Perfective aspect. The perfective aspect is elided before $S(\texttt{i})$, so we can resolve a bunch of entries with the rule

xix) $Q(\texttt{a})S(\texttt{i}) \to Q(\texttt{a})A(\texttt{wu})S(\texttt{i}), Q(\texttt{ka})S(\texttt{i}) \to Q(\texttt{ka})A(\texttt{wu})S(\texttt{i}), Q(\underline{\texttt{x}}'\texttt{a})S(\texttt{i}) \to Q(\underline{\texttt{x}}'\texttt{e})A(\texttt{wu})S(\texttt{i}),$
$Q(\texttt{ji})S(\texttt{i}) \to Q(\texttt{ji})A(\texttt{wu})S(\texttt{i}), Q(\texttt{tu})S(\texttt{i}) \to Q(\texttt{tu})A(\texttt{wu})S(\texttt{i}), A(\texttt{e})S(\texttt{e}) \to Q(\texttt{a})A(\texttt{wu})S(\texttt{i}).$

The last of these transformations handles only a single odd case.

Aspect `n-` prefix. A couple of problems arise with the irrealis and modality prefixes in the vicinity of the $A(\texttt{n})$ prefix. First, we guess the existence of an irrealis prefix with

xx) $A(\texttt{n}) \to R(\texttt{u})A(\texttt{n}),$

then we insert a missing modality prefix with the transformations

xxi) $A(\texttt{n})S(\texttt{yi}) \to A(\texttt{n})M(\underline{\texttt{g}})S(\texttt{yi}), A(\texttt{n})S(\texttt{du}) \to A(\texttt{n})M(\underline{\texttt{g}})S(\texttt{du}).$

Irrealis cleanup. The last few rules have to do with the pesky irrealis prefix, and are ever so slightly more "destructive". First, we have the rule

xxii) $Q(\texttt{a}) \to Q(\texttt{a})R(\texttt{u}), Q(\texttt{ka}) \to Q(\texttt{ka})R(\texttt{u}), Q(\underline{\texttt{x}}'\texttt{a}) \to Q(\underline{\texttt{x}}'\texttt{e})R(\texttt{u}),$
$Q(\texttt{ji}) \to Q(\texttt{ji})R(\texttt{u}), Q(\texttt{tu}) \to Q(\texttt{tu})R(\texttt{u}),$

which dishes out irrealis prefixes wholesale. To counter the possible doubling of irrealis prefixes in the previous rule, we have the rule

xxiii) $R(\texttt{e})R(\texttt{u}) \to R(\texttt{u}), R(\texttt{i})R(\texttt{u}) \to R(\texttt{u}), R(\texttt{o})R(\texttt{u}) \to R(\texttt{u}), R(\texttt{u})R(\texttt{u}) \to R(\texttt{u}),$

which also takes the opportunity to delete any funky renditions of this prefix. We also try adding irrealis prefixes before the modality prefix:

xxiv) $M(\underline{\texttt{g}}) \to R(\texttt{u})M(\underline{\texttt{g}})$

Some of these rules undoubtedly make a big mess of things, so we will include a rule that deletes an irrealis prefix, no questions asked:

xxv) $R(\texttt{u}) \to \epsilon$

We include a rule

xxvi) $Q(\texttt{a})A(\texttt{wu})S(\texttt{i}) \to Q(\texttt{a})R(\texttt{u})S(\texttt{i}), Q(\texttt{ka})A(\texttt{wu})S(\texttt{i}) \to Q(\texttt{ka})R(\texttt{u})S(\texttt{i}),$
$Q(\underline{\texttt{x}}'\texttt{e})A(\texttt{wu})S(\texttt{i}) \to Q(\underline{\texttt{x}}'\texttt{e})R(\texttt{u})S(\texttt{i}), Q(\texttt{ji})A(\texttt{wu})S(\texttt{i}) \to Q(\texttt{ji})R(\texttt{u})S(\texttt{i}),$
$Q(\texttt{tu})A(\texttt{wu})S(\texttt{i}) \to Q(\texttt{tu})R(\texttt{u})S(\texttt{i})$

that repurposes the pattern we created in rule (xix), and for subjects prefixes not preceded by a `wu-`, we have the rule

xxvii) $S(\texttt{tu}) \to R(\texttt{u})S(\texttt{tu}), S(\texttt{du}) \to R(\texttt{u})S(\texttt{du}), S(\texttt{i}) \to R(\texttt{u})S(\texttt{i}),$
$S(\texttt{yeey}) \to R(\texttt{u})S(\texttt{yeey}), S(\texttt{yi}) \to R(\texttt{u})S(\texttt{yi}).$

A final oddity. There are two entries that require the creation of a $Q(\texttt{u})$ disjunct prefix, so our last rule is

xxviii) $S(\texttt{i}) \to Q(\texttt{u})S(\texttt{i}).$

4. An explicit sequence of rewrites

In the previous section we described twenty-eight rewrite rules and the claim is that these rules are enough to resolve every entry in the charts found in [3]. Given an entry, a naïve way to check which rules apply is to simply try every combination. However, because multiple rewrites may be needed and because composition of rewrite rules is not commutative ($f \circ g = g \circ f$ does not hold in general), for every k rewrites we would like to apply, there are $k!$ orderings that may all produce different results. So *a priori* there are

$$\sum_{k=1}^{28} k! \binom{28}{k} \approx 8.29 \times 10^{29}$$

possibilities to check. Assuming each check takes a billionth of a second (which is rather optimistic), running through all of these possibilities will take several orders of magnitude longer than the current age of the universe to terminate. Of course, there are numerous ways one might be able to reduce these possibilities and make enumeration work, but we will take a different approach.

A function $f : S \to S$ is said to be *invertible* if there exists a function $f^{-1} : S \to S$ such that for all $w \in S$, $f^{-1} \circ f(w) = 1_S$, the identity function on S. The following negative result shows that nontrivial transformations are not invertible.

Lemma A. *Let S be the set of all sequences of tagged strings as defined above. Let $T : S \to S$ be a transformation such that for all $w \in S$, $f(w)$ is obtained by replacing the first occurrence of $x \in S$ in w with $y \in S$. Then T is invertible if and only if T is the identity transformation (which occurs precisely when $x = y$).*

Proof. If T is the identity 1_S, then we may take $T^{-1} = 1_S$. For the “only if” direction, suppose that T is not the identity transformation; that is, x is a nonempty string and $y \neq x$. We note that $T(x) = T(y) = y$, so that T is not injective and cannot have a well-defined inverse. ▮

5. Analysis of rewrite rules

[In this section, I want to analyse what the global sequence of rewrites actually means for individual strings. It is expected that for a given string w, most rewrites do not actually affect w directly, though in certain cases we may have to try and undo the same rewrite multiple times to try different combinations. It might also be interesting to see distributional data related to this (but I will have to code it up first).]

6. Grammar induction

[May not actually include, depending on how robust the above results actually are. If I end up not doing this section, it will merge with the next section. I was thinking of including a section about the general problem of grammar induction in formal language theory and artificial intelligence. My program does not come anywhere close to proper grammar induction, but I would like to make comparisons, because the goals are largely aligned with ours. Our data just happens to be at a smaller-scale and it also comes pre-tagged. In some sense, the way our data is set up makes the problem a lot easier than the general problem of grammar induction. However, if we want to extend the program to “real-world” data such as verbs found in the text corpus, then we might have to fall back on tried-and-tested approaches found in the literature (which may stumble because of the paucity of our data).]

7. Further directions

[Related to the grammar induction section. Reveal all of the shortcomings of our naïve program and discuss what can be done to extend the algorithm to work on the verbs found in the corpus, not just the nicely formatted verbs found in the prefix charts.]

References

[1] Antti Arppe, Christopher Cox, Mans Hulden, Jordan Lachler, Sjur N. Moshagen, Miikka Silfverberg, and Trond Trosterud, “Computational modeling of the verb in Dene languages. The case of Tsuut’ina,” *Working papers in Athabascan Linguistics* Red Book (2017), 51–68.

[2] James A. Crippen, *Tlingit text corpus* (GitHub repository, `https://github.com/jcrippen/tlingit-corpus`, 2015).

[3] James A. Crippen, *Tlingit verb prefixes* (GitHub repository, `https://github.com/jcrippen/tlingit-verb-prefixes`, 2020).

[4] Mans Hulden and Sharon T. Bischoff, “An experiment in computational parsing of the Navajo verb,” *Coyote Papers* **16** (2008), 110–118.

[5] Andrew J. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,” *IEEE Transactions on Information Theory* **13** (1967), 260–269.