A Dictionary of Thompson: Example of Computerized Lexicography

Sharon Mayes
University of Hawaii

Introduction

During the last two years I have been a research assistant for Laurence C. Thompson's Computerized Salish Dictionary Project, at the University of Hawaii at Manoa. My role has been to input Thompson's material from the Thompson language into computer storage and to output the material in various printed listings. As a result of my involvement in this project, I have also begun field work on a dialect of Thompson.[1] This direct contact has contributed to my understanding of the dictionary work.

Although the dictionary will not be available for some time, enough work has been done to warrant some sort of progress report. In part, this paper does serve as a progress report, giving some indication as to the nature of the dictionary. It is also my intention to illustrate the ways in which Salish linguistics can benefit from a computerized approach. Thus, many of the actual details involved in this work will not be discussed here. Instead, I will concentrate on a few interesting examples of the problems raised and/or solved by the computerization of a Salish dictionary.

For the benefit of those who are interested, the computer we use at the University of Hawaii is an IBM System 370/Model 158. The system of programs we use is written in the SPITBOL program-

[1] I am grateful to the National Science Foundation and the Melville and Elizabeth Jacobs Research Fund (Whatcom Museum, Bellingham, Washington) for support of this research. I also wish to thank L. C. Thompson, for introducing me to Salish; Bob Hsu, Ann Peters, and others for developing a fine body of programs and materials for all stages of computerized lexicography; and Bob Hsu, Neil Olsen, Peter Siegel and Joe Shacter for their help.

ming language.[2] Data is entered via time sharing on an IBM 2741 terminal with telephone hook-up. See Bibliography for further references. A sample entry from the dictionary is listed in Appendix I.

Printing

The following discussion will be clarified by a brief description of how the dictionary is printed. The orthography we are using for Thompson is similar to that used for other Salish languages, and so it is essentially quite different from English orthography. As a result, the regular keypunch machine (input) and computer line printer (output) are inadequate for our purposes.

However, the time sharing option (TSO) enables us to print a dictionary with unusual characters. Through TSO a typewriter terminal is used to communicate with the computer by telephone (instead of other media such as cards). Since an IBM Selectric typewriter terminal is used, we can switch to a special type element for entering and printing the dictionary material.[3]

Alphabetization

One advantage of processing a dictionary by computer is that no problem arises if you type in entries in the wrong alphabetical order. A program can be designed to alphabetize the entries when the time is right. In fact, if you want to alphabetize English (or a language with an orthography similar

[2] Programming languages and defined programs are capitalized.
[3] Camwil 721M, designed by L. C. Thompson.

to that of English), a system sort utility program is already available which will do so. This is a straightforward task because the IBM System/370 has the following collating order:

> blank.[(+&$*);-/,%_]?:#@'="abcdefghijklmnopqrstuvwx
> yzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789

The situation with Thompson is more complicated because the orthography we use has 52 letters, and many of them are different from those listed above.

For some languages that have unusual characters and an alphabet which does not conform to the above collating order, there is an easy solution. That is, for the purposes of sorting temporarily replace the individual characters that do not fit with other values from the above sequence, and make any necessary adjustments. For example, suppose a language has the following alphabet: ʔ,a,b,č,d,e,g,i,k,l,m,n,ŋ,o,p,s,t,u. Changing these to conform to the computer's collating order is a minor problem: replace 'ʔ' with '*' or any character preceding 'a' in the collating sequence; replace 'ŋ' with 'o', and change 'o' and 'p' to 'p' and 'q' respectively; replace 'č' with 'c'.

Notice that this works only for a language that has fewer letters than English. However, Thompson has twice as many letters. In this case, the easiest thing to do is replace all the Thompson letters with 'a,b,c,d,...z,A,B,C,D,...Z'. This also allows you to keep track of all the letters and check the sort listing more easily.

There are several ways that values can be substituted by a SPITBOL program. In the former example, few changes are involved, so the best strategy is to use replacement statements. In other words, after a segment has been isolated, write "if...then" instructions, such as "if '?' then '*'". This method is not practical for Thompson because 52 replacement statements would be required. Another method is available, whereby a TABLE[3a] is constructed which contains the replacement values for each letter. The TABLE is then consulted at the appropriate point by the program. Once the TABLE has been entered, all that is necessary is that an effective, efficient pattern match (isolating segments) be written. The pattern match is simplified for Thompson by grouping together glottalized consonants, labialized consonants, and other consonants sharing a particular feature. The resulting function for Thompson alphabetization looks something like Figure 1.

Some problems still remain. Note that the stressed and unstressed vowels have not been considered. If these are included in the original function, we will get an incorrect output. That is, all forms with 'é' will precede those with 'e' (or vice versa). Instead of this we want a stressed form to precede an unstressed one only if the two forms are otherwise identical. This is also the case for other diacritics--hyphens, brackets, parentheses, and slashes--which were suppressed (ignored) in the first function. The solution is to write a second pattern match which will differentiate only those forms which are identical in the list of forms generated by the first function. For example:

3a. TABLE is a SPITBOL function.

```
   H   TABLE(52)
  H['?'] = ['a'];  H['ʼ'] = ['b'];  H['a'] = ['c'];  H['c'] = ['d'];
  H['ç'] = ['e'];  H['ċ'] = ['f'];  H['e'] = ['g'];  H['ə'] = ['h'];
  H['ẹ'] = ['i'];  H['ɣ'] = ['j'];  H['ɣ̓'] = ['k'];  H['h'] = ['l'];
  H['i'] = ['m'];  H['ị'] = ['n'];  H['k'] = ['o'];  H['k̓'] = ['p'];
  H['kʷ'] = ['q'];  H['k̓ʷ'] = ['r'];  H['l'] = ['s'];  H['l̓'] = ['t'];
  H['ɬ'] = ['u'];  H['L̓'] = ['v'];  H['m'] = ['w'];  H['m̓'] = ['x'];
  H['n'] = ['y'];  H['n̓'] = ['z'];  H['o'] = ['A'];  H['p'] = ['B'];
  H['p̓'] = ['C'];  H['q'] = ['D'];  H['q̓'] = ['E'];  H['qʷ']= ['F'];
  H['q̓ʷ'] = ['G'];  H['s'] = ['H'];  H['ṣ'] = ['I'];  H['t'] = ['J'];
  H['t̓'] = ['K'];  H['u'] = ['L'];  H['w'] = ['M'];  H['w̓'] = ['N'];
  H['x'] = ['O'];  H['xʷ'] = ['P'];  H['x̣'] = ['Q'];  H['x̣ʷ'] = ['R'];
  H['y'] = ['S'];  H['y̓'] = ['T'];  H['z'] = ['U'];  H['z̓'] = ['V'];
  H['ʕ'] = ['W'];  H['ʕ̓'] = ['X'];  H['ʕʷ'] = ['Y'];  H['ʕ̓ʷ] = ['Z']
     GLOT = 'ʼ'
     OPTGLOT = GLOT | ''
     SUB = '.'
     OPTSUB = SUB | ''
     OPTLAB = 'ʷ' | ''
     VEL = ANY('kqʕ') OPTGLOT OPTLAB
     VELFRIC = 'x' OPTSUB OPTLAB
     EL = 'ɬ'
     GLOTBL = ANY('clmntwyLzɣp') OPTGLOT
     DOT = ANY('cəis') OPTSUB
     OTHER = ANY('?ʼaehou') | '(n)' | '(s)' | 's/'
     CRIT = ANY('()[]·/'"'")
 PHON = VEL | VELFRIC | EL | GLOTBL | DOT | OTHER | CRIT
 DEFINE('HANDLE(LINE)')                                    :(EOHANDLE)
HANDLE  LINE POS(0) PHON . SEG                             :F(FLAG)
 HANDLE = HANDLE H[SEG]                                    :(HANDLE)
FLAG  LINE LEN(1)                                          :F(RETURN)
 HANDLE = ' ' HANDLE
EOHANDLE
```

<u>Figure 1</u>. Function for Thompson Alphabetization.

First Substitute	Second Substitute	Thompson Form	Principle
a10	a101	ʔíx̣	stressed precedes unstressed
a10	a102	ʔix̣-	
dgd	dgd1	céc	unhyphenated precedes hyphenated
dgd	dgd2	c-é[c]-	
g	g1	e	suffix follows full form, stressed precedes unstressed
g	g3	-e	
g	g2	-é	
ga	ga3	-eʔ	prefix first, infix second, and suffix last
ga	ga1	[eʔ]	
ga	ga2	eʔ-	

If this is done correctly, only those forms which are truly identical (such as homonyms) will still have identical substitute forms, and they will remain in the same relative order in which they were entered. It should be noted that programming these close distinctions is difficult. In some cases it is probably easier to simply re-type the entries in their correct order.

Although the development of this program may seem like a bit of trouble, the result is definitely worth the effort. Several purposes are accomplished--1) the data can be alphabetized at any stage of work, so efforts can be concentrated on the actual writing of the dictionary entries; 2) much possibility for human error is eliminated through mechanical alphabetization according to regular principles; 3) other errors, such as typos, can be detected in checking through the list of substitute forms. It may also be noted that a system sort is an inexpensive process for the computer.

English Finderlist

The previous discussion illustrates the usefulness of mechanical consistency through computerization. Another major advantage is computer memory storage, which eases the burden involved in dealing with a large amount of data. Even though the data have been entered in a particular format, almost any subset of it can be extracted and used for other purposes. A striking example of this is the English-Thompson portion of our dictionary, which will be a listing (called a finderlist) generated from the Thompson-English data. The final version, of course, will be edited. However, a great deal of time will be saved because the original data have been formatted for English extraction. That is, when the English glosses for Thompson forms are entered, key words and phrases are prefixed with a * and the phrases are connected with _. A program has been written which will then extract the starred English words and phrases and print them as main entries, with the Thompson forms (for which they appear as glosses) listed underneath. The following are three sample entries from the English-Thompson dictionary.

animals
    capture wild *animals: kʷən·kʷén/-it, kʷən·kʷə́n/-it.
    eight (four-legged) *animals: piyeʔ/-ɬ/ʔú[ʔ]pn/kst.
    four *animals: mú[m]s.
    nine (four-legged) *animals: ,təm-ɬ/piʔ·éyeʔ.
    six (four-legged) *animals: Lá[L]qm/-kst.
    three (four-legged) *animals; keʔ[k]ɬé[ɬ]s.
    how many *animals?: kʷí[kʷ]nex.

[4]The size of the entire final dictionary (including English-Thompson) will be approximately 70,000 lines, 5,000 main entries, 23,000 subentries.

area
*area gets *dusty: n/ṗu[ʔ]ɬ/-úyṁxʷ.
*area, *country: tmíxʷ.
cold *area/*country: cɬ/-úyṁxʷ, tək ṭmíxʷ.
ground/*area gets *wet: ca[ʔ]qʷ/-úymxʷ.
high ground/*area, *up (on) the hill/mountain: xʔ/-úyṁxʷ.

away
avoid, get *away: kéw-ix.
avoid, get *away from someone: kéw-ix-m-s.
be in the distance, far *away: kʔéw.
come from far *away, from a *long *distance: kə·kéẇ/-xn.
discard, throw *away (because spoiled): pəL-t-és.
get *away from someone/something: ɬwéy-s.
go *away, *depart, *leave: nés.
go a long ways *away, go as *far as possible: keʔu/-xə́n.
remove, take *away: kéw-e-s.
retreat, withdraw, go (further) *away: kéw-ix.
run *away with something, *run *off with (something):
tʕʷ-i(y)xʷ/-ús-e-s.
take something far *away: keʔu-s-t-és.

Note, too, that the glosses can be checked for consistency, which is one of the otherwise frustrating aspects of dictionary work.

Linguistic Analysis

This principle of extraction can also be used as an aid for further linguistic analysis. The possibilities are quite broad, so only a few examples will be outlined here.

The type of information which can be pulled out depends to a certain extent, as we have seen, on how the data were originally formatted. The entries for our dictionary contain two major types of information--main entry or subentry type, and descriptive comments pertaining to the particular entry, such as underlying form or English gloss. In the format we use lines are called <u>bands</u>, so these categories of information are called <u>band names</u> and each line is coded with a band abbreviation.

The band names are listed with their abbreviations in Figure 2.

This format makes it a simple task to pull out a listing of the content of a particular category for all entries. For example, the ux (unexplained) band was created with this in mind. In the pre-final stages, all forms with ux can be listed for the purposes of further analysis and clarification. Some examples of ux comments:

```
.sf    k̓ʔéy̓
       g    *die_away, *disappear [of noise]
       ux   somehow related to k̓éynm-, listen?

..if    p̓e[p̓]kʷ-ût
        g    be *afloat, in the water [of boat]
        dl    Lytton
        ux    formation uncertain, reflexive?

.sf    p̓ə́k̓t
       g    *mica
       ux   appears to be an immediate form from a root *p̓ək̓,
            not otherwise recorded
       cr   pʔék, peʔk-, glitter

..ds    nek̓/-éw̓ɬ
        gc    glottalizing stem
        t    change money
        ux    probably /-éwiɬ with specializing extension

...df   máʕ/-xe-tn
        g    *moon, *month
        ux    presumably //máʕ-xə̆n-tən, but semantic development
              unclear

..df    mʔ-éstm
        g    polite address to person of opposite sex, primarily
             man speaking to woman
        ux    perhaps a fusion of mʔé-m and (s)/ʔéstm, in-law of
              opposite sex in same generation
```

Main Entry

.sf	surface form
.ss	surface stem
.pcl	particle
.af	affix
.lx	lexical suffix

Subentry

..bf	basic form
..if	inflected form
..is	inflectional stem
..df	derived form
..ds	derived stem
..bdf	basic derived form
..cs	compound stem
..ie	idiomatic expression

Paragraph Bands

r	root
uf	underlying form
**	grammatical limitation, rejected form
gc	grammatical code
g	gloss
p	probably, presumable gloss
t	tag
acl	acculturation development, adaptation
alt	alternate form
bot	botanical term (scientific name)
bsk	basketry term
eth	term of special ethnographic interest
irg	irregular form
kt	kin term
mth	mythology
pln	place name
psn	personal name
zt	zoological term (scientific name)
lo	loanword (source given)
dl	dialectal limitation
ux	unexplained form, problem
cr	cross reference

Figure 2. Information Band Names.

Thompson forms may also be examined for a specific phonological configuration, in order to provide examples for a grammar or to test a linguistic hypothesis. Furthermore, this is how a list of main entries for most lexical suffixes will be generated. Derived stems (already typed in) will be searched for lexical suffix initial boundary '/-'. If the form contains '/-', then what follows will be listed as a lexical suffix. For instance:

..ds	c̓ew̓/-ewíɬ t wash canoe	→	.ls	/-ewíɬ t canoe
..ds	c̓ew̓/-iʔs(t) t wash stone	→	.ls	/-iʔs(t) t stone
..ds	c̓ew̓/-kn̓ t wash back	→	.ls	/-kn̓ t back
..ds	c̓ew̓/-leʔ/-xn t wash extended foot	→	.ls	/-leʔ t extended
		→	.ls	/-xn t foot
..ds	c̓əq̓/-xə́n t throw-hit foot	→	.ls	/-xə́n t foot

Not only will this save a lot of time, but it will also be more exhaustive than if done by hand, since all of the various forms taken by each lexical suffix will be found in the computer's search. To anyone familiar with Salish linguistics, the advantages of such a listing are immediately clear.

The dl (dialect area) and lo (loanword) bands are useful for comparative linguistic work. For instance, a separate listing of forms with dl information will save a lot of work for the linguist who wants to check this data in the field. The extraction of an entire entry once it has been found to contain a certain band is somewhat more complicated for the programmer, but not a difficult problem.

Conclusion

A few examples have been discussed which clearly illustrate that the Thompson Computerized Dictionary Project represents a major development in Salish linguistics. The advantages of computerization have been mentioned, but it must be emphasized that the dictionary is only as good as its data. Credit for thorough data, organization, and insight into the capabilities of the computer goes to the creators of the dictionary, to their informants, and to the creators of the dictionary processing programs. It takes a great combined effort to produce such a useful body of data.

Appendix I

Sample entry from Thompson-English portion of dictionary.

.ss c̓éw̓
r c̓éw̓
t wash

..if c̓éw̓-e-s
g *wash something

..if n/c̓éw̓-e-s
g *wash inside of something
acl *wash *dishes

..if n/c̓ew̓-m
acl *wash *dishes, clean utensils after a meal

...if n/c̓éw̓-m-s-c
acl make someone wash dishes

...df n/c̓éw̓-mn
acl *dish-*pan, *dish-*water

...df s/c̓ew̓-mn
acl *gold *washings, coarse material eliminated
when panning for gold

..ds c̓ew̓/-enís
t wash tooth

...if n/c̓éw̓/-enís-m
g *wash one's *teeth

...df n/c̓ew̓/-enís-tn
acl *toothpaste

..ds c̓ew̓/-ewíł
t wash canoe

...if c̓ew̓/-ewíł-m
acl *wash one's *boat/*car

...if cuʔ·c̓ew̓/-ewíł-e-s
g wash plural boats/cars

..ds c̓éw̓/-iʔs(t)
t wash stone

...if c̓éw̓/-iʔs-e-s
g *wash (outside of) *basket

..ds c̓éw̓/-kn̓
t wash back

...if n/c̓éw̓/-kn̓-s
g *wash someone's *back

..ds c̓éw̓/-kst
t wash hand

...if c̓éw̓/-kst-m
g *wash one's *hands

..ds c̓éw̓/-leʔ/-xn
t wash extended foot

...if c̓uʔ·c̓éw̓/-leʔ/-xn-me
g *wash (both) one's *feet

..ds c̓éw̓/-ɬc̓iʔ
t wash body

...if c̓éw̓/-ɬc̓iʔ-s
g *wash *body of something [fish]

...if n/c̓éw̓/-ɬc̓iʔ-s
g [wash inside of body]

....ie n/c̓éw̓/-ɬc̓iʔ-s tə ṣúp
g administer an *enema to someone

..ds c̓éw̓/-ɬmx
t wash water-basket

...if c̓éw̓/-ɬmx-e-s
acl *wash *bottle/*jar

..ds c̓éw̓/-s
t wash face

...if c̓éw̓/-s-m
g *wash one's *face

...if n/c̓éw̓/-s-e-s
acl *wash *window

...df c̓éw̓/-s-tn
eth face-washing material (shredded dried cottonwood bark, makes lather when wet and rubbed)
acl face *soap
alt c̓éw/-s-tn tək ṣúp

...df n/c̓éw̓/-s-tn
g basin to wash face in

...df s/c̓éw̓/-s-tn
g water one has washed one's face in

..ds cew/-xn
t wash foot

...if cew/-xn-me
g *wash one's *feet

..ds cew/-xʷck
t wash chest

...if cew/-xʷck-m
g *wash one's *chest

..ds cew/-yep
t wash floor

...bf cew/-yep
g *wash *floor

...ie s/cew/-yep tək siL/-qt
acl *Saturday

..ds cew/ ymxʷ
t wash land

...bf cew/-ymxʷ(-m)
acl *wash *earth/*soil (as in panning for gold)

BIBLIOGRAPHY

Dewar, Robert. 1971. SPITBOL, Version 2.0. Illinois Institute of Technology. Reproduced by R. E. Griswold at Bell Telephone Laboratories, Inc.: New Jersey.

Griswold, R. E. 1975. String and List Processing in SNOBOL4. Prentice-Hall: New Jersey.

Griswold, R. E., J. F. Poage, I. P. Polonsky. 1971. The SNOBOL4 Programming Language. Bell Telephone Laboratories, Inc. Prentice-Hall: New Jersey.

IBM System/360 Operating System. Job Control Language Reference. Form GC28-6704-2. International Business Machines Corporation, 1972.

IBM System/360 Operating System. Time Sharing Option Command Language Reference. Form GC28-6732-2. International Business Machines Corporation, 1971.

Operating System Utilities. Form GC28-6586-15. International Business Machines Corporation, 1973.

Thompson, L. C. and M. T. Thompson. Thompson. Ms. University of Hawaii.